Users Guide

E20 Gateway (14.04)

351 Electronics Blvd. SW // Huntsville, AL 35824 // (877) 982-7888 // synapsewireless.com

CONTENTS

The **SNAPconnect E20** combines a Synapse **SM220 Series** RF module and an embedded Linux-based computer to provide connectivity (Ethernet, Wi-Fi, or serial) and site aggregation capabilities for **SNAP**-powered IoT networks across industrial temperature ranges.

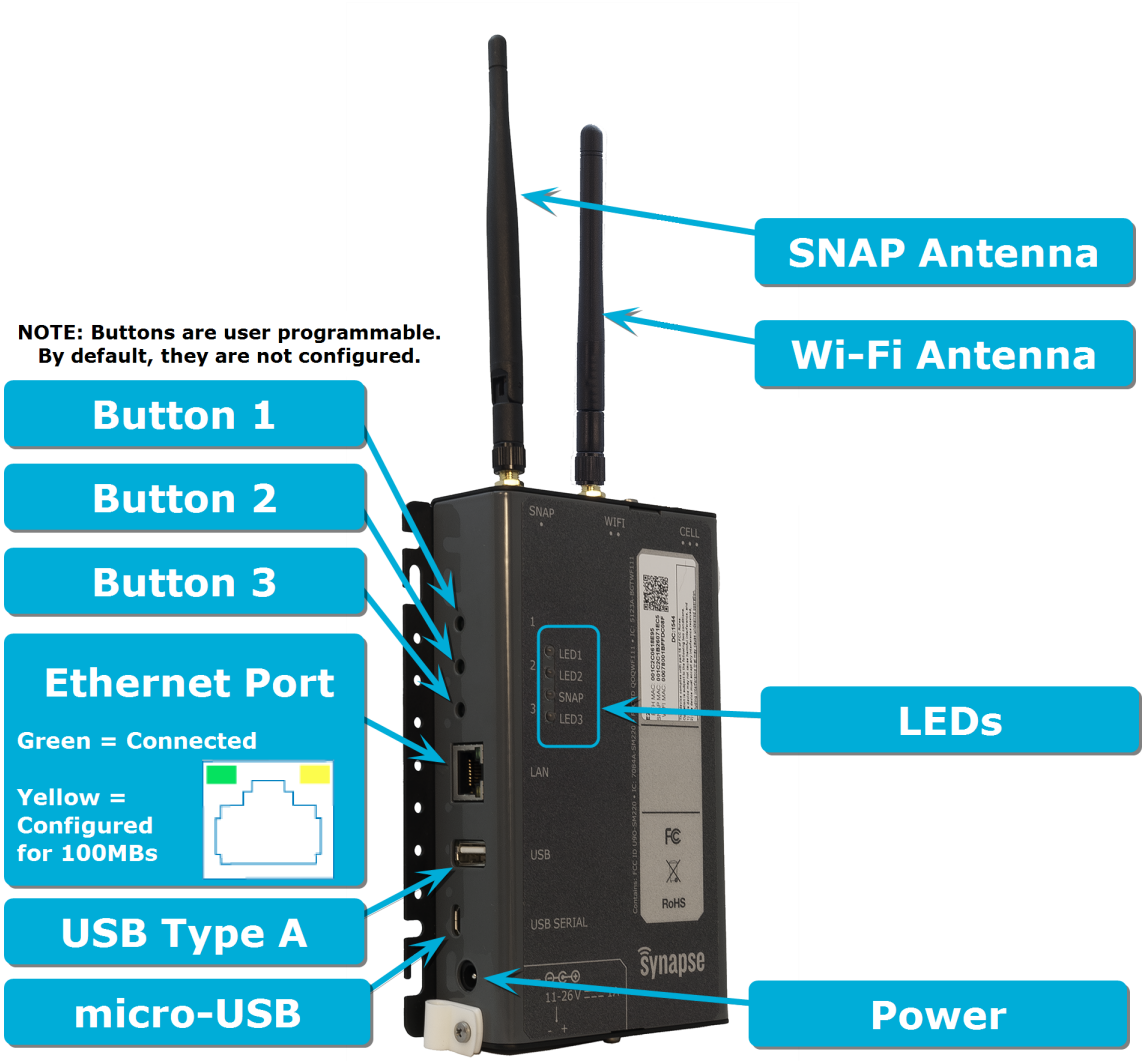
TCP/IP connections can bridge remote devices running **SNAP** into one common network, an effective method for centralizing data storage, performing web-based analytics, and monitoring remote applications.

Topics

CHAPTER

ONE

HARDWARE SETUP



1.1 Connecting the Antennas

The **SNAPconnect E20** will have a few antennas, as shown above. You can determine which antenna should go on which RP-SMA connector on the **SNAPconnect E20** based on the number of dots under the label on the gateway (one for SNAP, two for Wi-Fi) matching the number of dots marked on the included antennas.

Warning

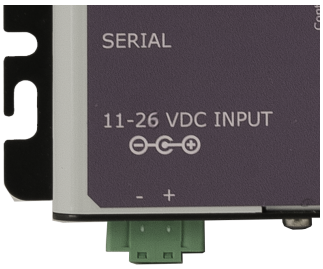
The E20 is an ESD sensitive device. Please use proper ESD precautions when handling the **SNAPconnect E20**.

When the antenna is not connected, it is recommended to attach a 50 OHM Terminator plug RP-SMA (part number 132360RP from Amphenol).

When it is time to attach the antenna, touch a grounded surface, remove the 50 OHM Terminator and screw on the antenna tightly. Do not overtighten or the RF pin in the bulkhead will crack, creating poor RF link quality.

1.2 Powering the E20

The E20 is powered by DC sources with output voltage in the range of 11V–26V and supplying at least 1A. The E20 has two provisions for supplying the DC input to the unit:



- The 2.0mm center-positive mono jack connector at the bottom of the left side
- The green terminal block connector on the bottom of the unit

The 2.0 mm center-positive mono jack connector on the side of the unit is typically used when powering the E20 using an external AC/DC power adapter. When powered by the mono jack, the DC input connector on the bottom of the unit is physically disconnected internally.

Use only UL 60950-1 2nd Edition AM1+AM2 Certified LPS Power Supplies, rated for the relative environmental conditions suited to your use location with an output of 11-26VDC, 1A Max. If you're applying power via the terminal block, the positive and negative terminals are marked on the front label of the unit (see image to the right). The terminal block header that connects to the E20 requires 14-30AWG wire. (The terminal block header is part #1900882 from Phoenix Contact. See their website for more info.)

The DC source should be a SELV source in accordance with IEC 60950-1, properly certified by the relevant authorities in your location, having ratings suitable for the environmental conditions of the installation.

Warning

All DC wiring should be done in accordance with all relevant local and national wiring regulations and should be protected by a suitably rated fuse or circuit breaker sized according to the relevant wiring codes. The maximum interrupting rating of the overcurrent protective device is not to exceed 8A.

Use only AC/DC power adapters that are properly certified by the relevant authorities in your location, having ratings suitable for the environmental conditions of the installation.

Note

If the E20 is installed into an end product, wiring should be done in accordance with the relevant product safety standard of the end product.

CHAPTER

TWO

GETTING STARTED

These directions provide the steps for terminal access to your **SNAPconnect E20** from either Windows, Linux or macOS.

2.1 Requirements

- DC power; the **SNAPconnect E20** requires 11 to 26 volts DC from either the barrel or terminal-block connector.
- A standard USB to Micro-USB cable, such as might be used for charging a cell phone.
- Terminal emulation software such as Tera Term, PuTTY, cu, screen, etc.

2.2 Connect USB to Host PC

You will need to connect the USB cable to the **SNAPconnect E20** and determine which serial port was assigned. This process is different based on the OS.

> **Note**
>
> If you find that your host PC cannot connect to the **SNAPconnect E20** over the USB connection, you may need to install the Silicon Labs CP210x USB to UART VCP drivers.

2.2.1 Windows

Check under “Ports” in the Device Manager and look for “Silicon Labs CP210x USB to UART Bridge (COMxx)”. The “COMxx” will indicate the serial port assigned (e.g., COM3, COM88).

2.2.2 Linux

The USB connection will be in the `/dev` directory. Check the directory before and after connecting the USB cable to identify the new `ttyUSBx` where `x` indicates the USB connection assigned (e.g., ttyUSB0). For example:

```
$ ls /dev/ttyUSB*

  /dev/ttyUSB0
  /dev/ttyUSB1   # <- new device
```

2.2.3 macOS

The USB connection will be in the `/dev` directory. Check the directory before and after connecting the USB cable to identify the new device. It is typically `/dev/tty.SLAB_USBtoUART`.

```
$ ls /dev/tty.*

  /dev/tty.Bluetooth
  /dev/tty.SLAB_USBtoUART # <- new device
```

2.3 Terminal Access

Use a terminal emulator to communicate to the **SNAPconnect E20**'s serial port using the following serial port settings:

- 115200 baud
- 8 bits
- No parity
- 1 stop bit
- No flow control

For example, using cu in **Linux**:

```
$ sudo cu -l /dev/ttyUSB0 -s 115200
```

Another example, using screen in **macOS**:

```
$ sudo screen /dev/tty.SLAB_USBtoUART 115200
```

2.4 Login

Login to your **SNAPconnect E20** gateway the first time using the default credentials:

- Username: snap
- Password: synapse

> **Note**
>
> You must change your password the first time you log in.

2.5 Connect to the Internet

The easiest way to do this is to make a wired connection to an Internet-connected router that supports DHCP. If you wish to configure your device for a static IP address or configure your Wi-Fi at this point, you may do so via the serial connection before making your Internet connection.

2.6 Update Python Development Libraries

Before starting work with the **SNAPconnect E20**, you'll want to make sure you have the latest versions of the software installed on the unit. The gateway makes use of Python Development Libraries that are sometimes updated to add new functionality and correct bugs. You can update the version installed on your unit by running the command:

```
sudo apt-get update
sudo apt-get install python-dev
```

2.7 Set the Clock

First, you should specify the timezone in which your device will reside. An easy way to do this is to use `tzdata`, which allows you to select the general region, and then select the specific zone for your location. You can run it with the command:

```
sudo dpkg-reconfigure tzdata
```

Next, and only if the gateway's date is not set (i.e. it is not connected to a network, so it does not set the date from an NTP server, and the hardware clock has never been set), set the date manually. The following example sets the date to `April 20, 2017, at 12:30:59 p.m.`

```
sudo date --set "2017-04-30 12:30:59"
Sat Apr 30 12:30:59 CDT 2017
```

You can set the hardware clock from the system clock using the `hwclock` command.

```
sudo hwclock -wu
```

2.8 Update SNAPconnect

The SNAPconnect software enables the connection from your **SNAPconnect E20** device to the rest of your SNAP-powered network.

To update SNAPconnect, type the command:

```
sudo -H pip install --extra-index-url https://update.synapse-wireless.com/pypi/ --
↪upgrade snapconnect
```

2.9 Update PyCrypto

The PyCrypto project is required for using AES128 encryption on your radio network.

To update PyCrypto type the command:

```
sudo -H pip install --extra-index-url https://update.synapse-wireless.com/pypi/ --
↪upgrade pycrypto
```

That's it! Your **SNAPconnect E20** is now ready to work with your SNAP-powered network. Your Python program, using the **SNAPconnect** library, or the **SNAPtoolbelt** utilities, interfaces with the gateway's SNAP module directly, and the rest of your nodes through that. You can also have full Internet access (either through a wired connection or a Wi-Fi connection).

You can find examples of other people's efforts on the Synapse Wireless repository at GitHub: https://github.com/synapse-wireless.

The site includes sample projects for things like sending data collected by SNAP-powered nodes to cloud services, or a gateway-hosted web server. Download the code there, or fork it for your own projects. Better yet, contribute to the code base for other users.

CHAPTER

THREE

E20 SOFTWARE SPECIFICS

The **SNAPconnect E20** uses Canonical's Ubuntu 18.04 as of v2.0.0, running a custom Linux kernel based on the i.MX6 kernel by Freescale. Older versions of the E20 OS are based on Ubuntu 14.04. There are many resources out there for learning about Ubuntu online, and the topic possibilities far exceed the scope of this manual; however, there are a few details that warrant discussion.

3.1 Passwords and root access

The default configuration for Ubuntu Linux is to have the root user disabled. This is a security precaution, as it means a hacker who comes across a connection to your device does not automatically know the login name of a user with full administrative rights to your device.

Instead, Ubuntu works with the sudo paradigm: when you need to perform a function that requires administrative access, you preface your command with sudo and are prompted for your password (as a reminder that what you are doing could potentially affect the device's ability to function).

The default snap user on the E20 has sudo access, and can perform all device administrative tasks. You can create user accounts on the device and grant them sudo access as well. For added security, you can create sudo-enabled user accounts and then delete the snap user account to further reduce a potential hacker's knowledge of how to access the gateway.

If you would rather work with the root account anyway, you can enable it by assigning a password:

```
sudo passwd root
```

Similarly, you can change the snap password with the same command:

```
sudo passwd snap
```

Note

No account can connect via SSH without a password, though connecting over a serial terminal session is possible for accounts with no password.

3.2 E20 Software Packages

The **SNAPconnect E20** comes with several support packages installed, and additional ones are available via apt and pip.

You can pull the latest aptupdates for your gateway by calling:

```
sudo apt-get update
sudo apt-get upgrade
```

You can upgrade SNAPconnect, SNAPtoolbelt, and the encryption libraries necessary for AES128 communications using these commands:

```
sudo -H pip install --extra-index-url https://update.synapse-wireless.com/pypi --upgrade␣
↪snapconnect snaptoolbelt pycrypto
```

CHAPTER

FOUR

LEDS AND BUTTONS

The **SNAPconnect E20** includes three tri-color LEDs that you can control from your programs, plus three buttons you can monitor.

4.1 LEDs

Each of the three LEDs can be red, green, or amber. Each has a script you can use to set the LED state:

```
led-a red
led-b green
led-c amber
led-a off
```

You do not need to use sudo to control the LEDs. You can simply add the users you want to have control of the LEDs to the `leds` group.

By default, all three of these LEDs will turn amber when the gateway is powered on and then turn off after the device boots.

Each of the three LEDs is controlled by a pair of GPIOs from the i.MX6 processor, with one controlling the red, one controlling the green, and the two of them together generating amber.

If you would rather control the LEDs using the GPIOs rather than the scripts, these are the lines for each LED and color:

GPIO 40	LED-1	red
GPIO 41	LED-1	green
GPIO 42	LED-2	red
GPIO 43	LED-2	green
GPIO 44	LED-3	red
GPIO 45	LED-3	green

4.1.1 SNAP Module-Controlled LED

The unit's SNAP module controls the tri-color LED labeled "SNAP" via `GPIO_A4` (green) and `GPIO_A5` (red). (For amber, use both green and red.) This LED is only accessible via the SNAP module. It cannot be controlled by the E20's i.MX6 processor, except through calls to the SNAP module.

These two IO lines from the SNAP module will light their respective colors when written high. This sample code demonstrates its use:

```
from synapse.platforms import *

@setHook(HOOK_STARTUP)
def onStartup():
    setPinDir(GPIO_A4, True)
    setPinDir(GPIO_A5, True)
    LED_off()

def LED_off():
    writePin(GPIO_A4, False)
    writePin(GPIO_A5, False)

def LED_green():
    writePin(GPIO_A4, True)
    writePin(GPIO_A5, False)

def LED_red():
    writePin(GPIO_A4, False)
    writePin(GPIO_A5, True)

def LED_amber():
    writePin(GPIO_A4, True)
    writePin(GPIO_A5, True)
```

The SNAP LED is the only LED controllable directly from the SNAP module. The other three LEDs are controlled from the E20's i.MX6 processor.

4.2 Buttons

The three buttons on the left side of the **SNAPconnect E20** are fully user-accessible. There are button commands that print the button status to STDIO and return the button status (as 1 for up or 0 for pressed):

```
button-1
button-2
button-3
```

You can monitor the i.MX6 processor GPIOs directly rather than using the Bash scripts if you find that to be easier:

GPIO 117	BUTTON-1
GPIO 118	BUTTON-2
GPIO 119	BUTTON-3

CHAPTER

FIVE

WORKING WITH THE SNAP MODULE

The **SNAPconnect E20** contains a Synapse Wireless SM220 surface-mount SNAP module, which it can access via serial ports `/dev/snap0` and `/dev/snap1` connected to `UART0` and `UART1` on the module, respectively. By default, SNAP-powered modules communicate serially over `UART1`, so when making your **SNAPstack** or **SNAPtoolbelt** connection to the module, you should use `/dev/snap1` unless you have modified your modules's default UART settings.

For detailed instructions on **SNAPstack**, please consult the SNAPstack Users Guide.

In addition to the serial connections, there are two GPIO pins from the gateway that are tied to lines on the SM220 for controlling and signaling.

E20 Pin	SM220 Pin	Purpose
GPIO 33	GPIO_F1	You can use this pin as a signaling semaphore or to wake the SM220 when it is sleeping.
GPIO 34	RESET	You can use this pin to reboot the SM220.

By default, the **SNAPconnect E20** ships with **SNAPtoolbelt** installed. **SNAPtoolbelt** can be very handy when needing to perform simple operations or maintenance on the SNAP module.

5.1 Waking the SNAP Module

At times it may be helpful to have the SNAP module in your E20 sleep, and then be woken by the E20's processor. You can easily do this by defining `GPIO_F1` on the SNAP module as a wake pin, like this SNAPpy script:

```
from synapse.pinWakeup import *
from synapse.platforms import *

@setHook(HOOK_STARTUP)
def onStartup():
    setPinDir(GPIO_F1, False)
    setPinPullup(GPIO_F1, True)
    wakeupOn(GPIO_F1, True, True)
```

To wake the SNAP module you will need to pull the E20s `GPIO 33` high, pause a second, and then pull the line low.

5.2 Resetting the SNAP Module

Just as you can wake a sleeping SM220, there is a pin you can use to reset your module should you need to.

Invoke this Bash script to briefly pull the reset pin low and then release it to high, resetting the node:

```
/usr/local/bin/reset-snap-node
```

5.3 Recovering the SNAP Module

Several things can make a module unresponsive, including: setting an encryption key that you then forget, or a script that sends the node to sleep with an invalid wake pin defined, or even a script that sends the node into an infinite loop.

SNAPtoolbelt provides help here, with the `snap recover` options.

If you find that your SNAP module is unresponsive or unreachable over the air or serially, the first suspect is typically the user script on the module. So, the first thing to try in module recovery is forcibly removing the SNAPpy script from your SNAP module:

```
snap recover erase-script SM220UF1
```

This leaves your SNAP module's NV parameters untouched, but removes the existing SNAPpy script from the node. You can then load an appropriate script over the air or serially.

If this does not restore your access to the node, the most likely reason for your inability to communicate is mismatched configuration (NV) parameters on the node. This could be the result of different encryption keys or encryption types, misconfigured UARTs, differences in how many CRCs are expected, or some other configuration setting. The easiest thing to do next is to have the node default its NV parameters:

```
snap recover default-nvparam SM220UF1
```

This clears the encryption settings (no key, no encryption), sets UART connections to their default settings (UART1, 38,400 baud, 8N1), and clears other settings to their default levels. (Refer to the SNAP Reference Guide for other defaults.)

Typically it is best to start with clearing the script in your node before resetting its parameters, because it is possible for the script to reset (away from default values) parameters that you just reset (to default values).

Note

Resetting your SNAP module to default settings does not automatically mean that other devices can talk to it, over the air or serially. It does mean that you now know how to configure those devices to talk to it. If you have another radio device on channel `13` using network ID `0xABCD`, you will have to set that device to channel `4`, network ID `0x1C2C` to talk to your defaulted SNAP module. You can then use that radio connection to move the gateway's SNAP module to your preferred network settings. Or, you could change those settings serially from your **SNAPconnect E20** — if your gateway is set to communicate serially the way that your SNAP module is (considering encryption keys and types, serial rates, etc.). The point is: defaulting a device doesn't mean you have it where you want it, only that you now know where to go look for it.

5.4 Controlling the E20 from the SNAP Module

Just as there are lines from the E20's i.MX6 processor to the **SM220 Series** as a wake pin and a reset, there are corresponding pins back to the i.MX6 from the **SM220 Series**:

SM220 Pin	E20 Pin	Purpose
GPIO_F	GPIO 32	You can use this as a one-bit signal from the SM220 even when SNAPconnect software is not running on the E20.
GPIO_C	RESET	This pin is tied to the system reset line on the E20, active low. You can use this to reboot the i.MX6 processor without interrupting service on the **SM220 Series**.

The reset line will force a reboot of the E20's i.MX6 processor. This may cause the loss of unsaved data, and as with any uncontrolled shutdown of a computer it is not recommended that you use this often. It is intended only to recover an unresponsive E20. The following sample SNAPpy script demonstrates rebooting the i.MX6 processor from the SM220:

```
from synapse.platforms import *

@setHook(HOOK_STARTUP)
def on_startup():
    setPinPullup(GPIO_C4, True)
    writePin(GPIO_C4, True)
    setPinDir(GPIO_C4, True)

def resetE20():
    pulsePin(GPIO_C4, 1, False)
```

In contrast, the **GPIO_F2** pad on the **SM220 Series** can signal the **GPIO 32** line on the i.MX6 without any danger to either system, and the signal can be processed or ignored by your E20 program as you choose. The following sample Bash script demonstrates how to watch for the pin change from your gateway's perspective:

```
if [[ ! -d /sys/class/gpio/gpio32/ ]] ; then
    echo 32 > /sys/class/gpio/export
    echo in > /sys/class/gpio/gpio32/direction
fi
while [[ `cat /sys/class/gpio/gpio32/value` != 0 ]] ; do
    sleep 1
done
echo "Got interrupt!"
```

CHAPTER

SIX

WI-FI SETUP

By default, the Wi-Fi interface on the **SNAPconnect E20** is not active on startup.

In This Section

6.1 Enabling Wi-Fi

Edit the interfaces file at `/etc/network/interfaces` file using the editor of your choice.

Remove the # from the beginning of the following line:

```
#auto wlan0
```

On the next reboot, the Wi-Fi connection will automatically activate, though additional configuration is necessary for it to connect.

6.2 Connecting to an Access Point

Connecting to a access point using WPA encryption is fairly easy. You need to provide the `/etc/wpa_supplicant.conf` file with your desired network SSID and a passphrase key.

The easiest way to do this is to use the wpa_passphrase application:

```
sudo bash -c 'wpa_passphrase "myssid" "mypassword" >> /etc/wpa_supplicant.conf'
```

This command generates a passphrase key from your password and then appends the appropriate text to the `/etc/wpa_supplicant.conf` file.

After you do this, you should edit the `/etc/wpa_supplicant.conf` file to remove the line that includes the clear text of your passkey, and to make sure there are not issues with conflicting network entries. You may also need additional options, depending on your network setup. (Such configuration is beyond the scope of this document.)

You can now reboot (or use `ifup wlan0`) to bring up the interface and connect to the network.

6.3 Setting Up Access-Point Mode

You can configure your gateway to work as an access point for other Wi-Fi devices. This can be useful if you want to be able to connect directly to your gateway with a laptop or phone to administer your application.

6.3.1 Install udhcpd

Begin by making sure the udhcpd package is installed:

```
sudo apt-get install udhcpd
```

6.3.2 Set Your SSID and Passphrase

Generate your passphrase using the wpa_passphrase application. You can direct this output to a file for recall later, or track it in the method of your choice.

```
$ wpa_passphrase 'myssid' 'mypassword'
network={
    ssid="myssid"
    #psk="mypassword"
    psk=2f0568b3492812bd56b946dbaf3fd7dd669b9a4602a09aa6462ff057949b025c
}
```

6.3.3 Set up /etc/udhcpd.conf

The `interface` definition in the `/etc/udhcpd.conf` file defaults to eth0. Find the definition in the file (typically within the first dozen lines, but it may vary depending on your current configuration) and change it to wlan0 instead.

```
# The start and end of the IP lease block
start 192.168.0.20 #default: 192.168.0.20
end 192.168.0.254 #default: 192.168.0.254
# The interface that udhcpd will use
interface wlan0 #default:eth0
```

6.3.4 Set udhcpd to Run by Default

Edit the `/etc/default/udhcpd` file to change the `DHCPD_ENABLED` parameter from “no” to “yes”.

```
DHCPD_ENABLED="yes"
```

6.3.5 Assign a Static IP Address

Edit the `/etc/network/interfaces` file to assign a static IP address so the gateway can act as an access point. By default, the file contains this configuration text:

```
iface wlan0 inet dhcp
    wpa-conf /etc/wpa_supplicant.conf
    wpa-driver wext
```

You can either replace that text with the new configuration text, or comment those lines by inserting a # character at the beginning of each line, and then add your new configuration text:

```
iface wlan0 inet static
    address 192.168.0.1
    netmask 255.255.255.0
```

Specify the IP address you wish to use for your access point, `192.168.0.1` in the example above. You should choose an appropriate private network address, suitable for your needs.

6.3.6 Configure the LAN with iwpriv

Use the wpa_passphrase information you generated earlier as parameters for a call to iwpriv, specifying a channel and other parameters appropriate for your environment:

```
sudo iwpriv wlan0 AP_SET_CFG ASCII_CMD=AP_CFG,SSID="myssid",SEC="wpa2-psk",
↪KEY=2f0568b3492812bd56b946dbaf3fd7dd669b9a4602a09aa6462ff057949b025c,CHANNEL=1,
↪PREAMBLE=1,MAX_SCB=8,END
```

6.3.7 Start AP Mode

```
sudo iwpriv wlan0 AP_BSS_START
```

6.3.8 Stop AP Mode

```
sudo iwpriv wlan0 AP_BSS_STOP
```

CHAPTER

SEVEN

USING USB ACCESSORIES

The **SNAPconnect E20** has drivers to support many USB devices, such as a second SNAP-powered bridge (using an SN220 or SN132 carrier board), Wi-Fi devices, cell modems, or external storage. While the complete details of configuration options available on these devices fall outside the scope of this document, there are some common considerations that may prove useful.

7.1 USB Power

The USB 2.0 connection on the E20 is not rated as a "battery-charging" connection, and may not provide sufficient current for high-drain devices, such as some external hard drives. If you find you are having problems with your USB devices (e.g., external hard drives failing to mount, or cellular connections losing their connections), we recommend you try connecting the device to the E20 through a powered USB hub.

7.2 Connecting to an Additional SNAP Device

The E20 can support a second SNAP-powered node through its USB port. You can connect an SN132 or SN220 SNAPstick, or you can use an FTDI USB-serial cable to connect to an SN171 ProtoBoard or some other hardware that uses a DB9 connector to make an RS232 serial connection. This can allow your E20 to act as a bridge between two radio subnets, where radios are on some combination of different frequencies, different network IDs, and/or different channels.

Whichever device you use, plug the device into the E20's host USB port and (among other messages) you should see something similar to:

```
usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
```

or:

```
usb 1-1: cp210x converter now attached to ttyUSB0
```

The key here is the line that says `converter now attached to ttyUSB#`. You will use this device handle,``ttyUSB#``, to communicate with the SNAP device. In your SNAPconnect application, you would open a connection to the device like this:

```
com.open_serial(type=SERIAL_TYPE_RS232, '/dev/ttyUSB0')
```

7.3 Using usb_modeswitch

Many USB Wi-Fi and cell modems now come with a small amount of onboard storage, typically used to automatically install drivers when connected to a Windows host. When the device first connects, it appears as a small flash drive or virtual CD-ROM. After installing the necessary drivers, the Windows host sends a signal to the device instructing it to "mode switch" – to unmount the storage and expose itself as a Wi-Fi (or cellular) device. Ubuntu Linux also automatically handles many of these devices, but there may be some out there that Ubuntu does not recognize by default. If you find that the E20 is not recognizing your device, consider installing `usb_modeswitch`, which contains a library of parameters for converting devices like these.

```
sudo apt-get install usb-modeswitch
```

Then, plug your device in again and you are likely to find that it works as expected.

CHAPTER

EIGHT

ACCESSING THE MICROSD SLOT

The **SNAPconnect E20** includes a microSD slot for reflashing your device to its factory state. You can also use a card in this slot as additional flash storage on your gateway if you need it. The following instructions will work for ext4-, FAT32-, or exFAT-formatted cards. Ubuntu does not support exFAT by default. You will need to run the following command for exFAT support:

```
sudo apt-get install exfat-fuse exfat-utils
```

To access a card in the microSD slot:

1. **Insert the microSD card into the microSD card slot under the access cover on the rear of the E20:**
    - Slide the microSD card carrier toward the bottom of the unit (away from the antenna end) about a sixteenth of an inch (1.5 mm).
    - Open the card carrier frame. It is hinged at the top (antenna end) edge.
    - Insert your microSD card, contacts first, with the contacts exposed.
    - Close the card carrier frame, and slide it toward the top of the unit to lock the card in place.
2. Create a mount point for the card. In this example, the directory will be named sdcard, and it will be in the `/mnt` directory. (If you have previously done this, you do not need to repeat it.)

    ```
    sudo mkdir /mnt/sdcard
    ```

3. **Mount the card. (For these commands, replace `p1` with the partition number you want to mount.)**
    - For cards formatted with the ext4 file system:

        ```
        sudo mount -t ext4 /dev/mmcblk0p1 /mnt/sdcard
        ```

    - For cards formatted with the FAT32 file system:

        ```
        sudo mount -t vfat /dev/mmcblk0p1 /mnt/sdcard
        ```

    - For cards formatted with the exFAT file system:

        ```
        sudo mount -t exfat /dev/mmcblk0p1 /mnt/sdcard
        ```

You can confirm it is mounted by using the mount command and looking for an entry like the following (with the appropriate file system format):

```
/dev/mmcblk0 on /mnt/sdcard type ext4 (rw)
```

You can use the ls command to list the available partitions:

```
ls /dev/mmcblk0p*
```

CHAPTER

NINE

FACTORY RESTORE

Restoring an **SNAPconnect E20** to the factory default is accomplished using a microSD card image. Follow the steps below to restore your gateway.

9.1 Download image

Download the newest microSD card E20 installer image.

9.2 Write new image to a microSD card

9.2.1 Using Etcher (Windows, macOS, Linux):

Using Etcher, program the microSD card with the downloaded image.

9.2.2 Using dd (command line, Linux/macOS):

Use `dd` from the command line, where `if` is the sdcard image file and `of` is the device location of the microSD card device. For example:

```
dd if=e20-VERSION-sdcard.img of=/dev/sdX bs=1M

sync
eject /dev/sdX
```

Warning

Ensure that the `of` variable points to the correct microSD card device location before executing this command.

Depending on your system, you may need to use `sudo`.

9.3 Insert microSD

First, remove power to the **SNAPconnect E20**. Remove the access cover on the rear of the **SNAPconnect E20**, and insert the microSD card into the microSD card slot.

Slide the microSD card carrier toward the bottom of the unit (away from the antenna end) about a sixteenth of an inch (1.5 mm). Open the card carrier frame, which is hinged at the top (antenna end) edge. Insert your microSD card, contacts first, with the contacts exposed. Close the card carrier frame, and slide it toward the top of the unit to lock the card in place.

9.4 Use microSD to restore E20

Hold the button marked S4, apply power, and release the button.

LEDs 1, 2, and 3 will turn from amber to red when the restore process is in progress. This can take up to 5 minutes or more, depending on factors such as the speed of your microSD card.

The three LEDs will turn green, indicating that the process is complete. Disconnect power from the **SNAPconnect E20**, remove the microSD card, and reinstall the access cover.

Your **SNAPconnect E20** is now restored and ready to use.

CHAPTER

TEN

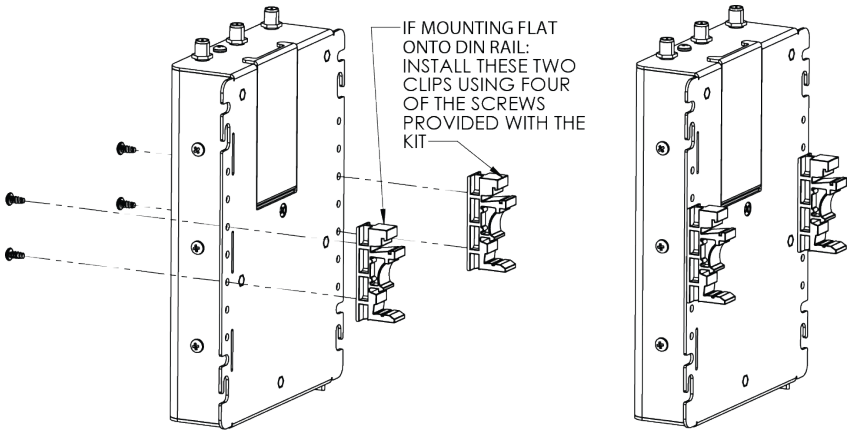
MOUNTING INSTRUCTIONS

The E20 has a number of mounting holes along the outside flanges of the unit, any of which can be used for mounting the E20 to a solid surface.

Synapse also provides an optional DIN Rail Mounting kit (part AC021-001) if you wish to mount the E20 on a DIN rail. There are two options for DIN rail attachment depending on your available space.

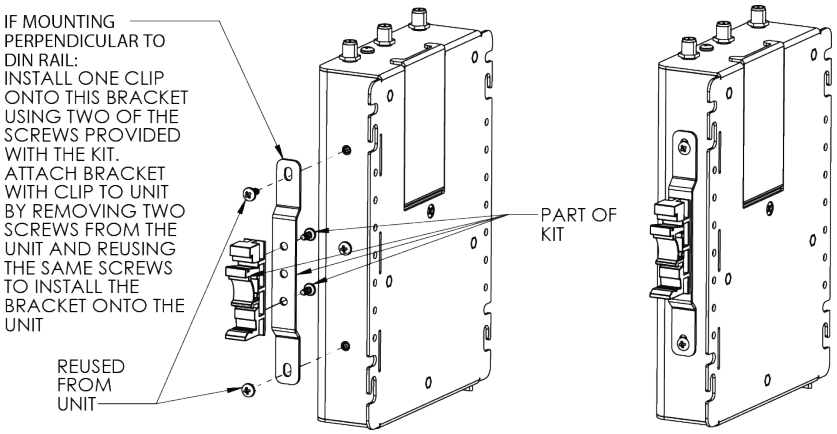
10.1 Mounting Flat Against the DIN Rail

Attach the two white mounting clips to the E20 as shown in the diagram below:



10.2 Mounting Perpendicular to the DIN Rail

1. Attach the white mounting clip to the stand-off bracket using the two screws provided with the DIN rail kit as shown in the diagram below.
2. Remove the top and bottom screws from the narrow edge of the E20, leaving the one in the center.
3. Using the screws you just removed, attach the mounting bracket to the E20 as shown in the diagram.


IF MOUNTING PERPENDICULAR TO DIN RAIL:
INSTALL ONE CLIP ONTO THIS BRACKET USING TWO OF THE SCREWS PROVIDED WITH THE KIT.
ATTACH BRACKET WITH CLIP TO UNIT BY REMOVING TWO SCREWS FROM THE UNIT AND REUSING THE SAME SCREWS TO INSTALL THE BRACKET ONTO THE UNIT
REUSED FROM UNIT
PART OF KIT

CHAPTER

ELEVEN

TECHNICAL SPECIFICATIONS

The Synapse Wireless E20 gateway is an ARM Cortex-A9 based Linux computer running Ubuntu. It incorporates a 2.4 GHz Synapse RF module that connects the device to SNAP-powered mesh networks. The gateway has Wi-Fi and Ethernet network connectivity, and serial connectivity is available through a Micro-USB connection. This arrangement provides for a wide range of possibilities for monitoring sensor networks, controlling remote devices, and driving the Internet of Things

11.1 Hardware Specifications

OS	Ubuntu 18.04 LTS for v2.0.0 and above Ubuntu 14.04 LTS for v1.2.3 and below
CPU	ARM Cortex-A9, 800MHz (Freescale i.MX6-S)
Flash	4GB eMMC
RAM	512M DDR3, 400MHz
Network	10/100 Ethernet, WiFi, SNAP 2.4 GHz
USB host	1 type A
USB client	1 Micro-USB - SiLabs CP2102
Min Operating Temperature	-40°C
Max Operating Temperature	70°C
Board Size	15.5cm x 9cm x 2cm
Unit weight	582 grams (without antennas)
Input Voltage	11-26V DC AC power supply sold separately. See synapsewireless.com for more information.
Storage Expansion	µSD
LEDs / Buttons	4 LEDs 3 Buttons

Note

When running an application that demands unusually intensive CPU/Memory resources at 70C, the temperature on the processor core might reach up to 90C resulting in performance degradation. For more information, see http://cache.freescale.com/files/32bit/doc/app_note/AN4579.pdf.

Note

This equipment is certified by Underwriters Laboratories for operation in a maximum ambient temperature of 65°C. The product safety standard to which this unit is evaluated and certified, IEC 60950-1, and UL 60950-1 for the US and Canada, specifies a maximum temperature limit of 70°C on metal surfaces that may be touched. During operation, there is a slight temperature rise on the surfaces of the E20. Therefore, when installed into an ambient environment at 70°C, the surface temperatures on the E20 may exceed the limit of 70°C.

11.2 Approved FCC/ISED Antennas

11.2.1 SNAP Antennas

Part Number	Type	Gain	Impedanc	Applica-tion	Min. Sepa-ration
Pulse W1027	Dipole (quarter-wave RPSMA)	3.2 dBi	50 Ω	Fixed/Mob	20 cm.
HyperLink HG2405RD-RSP	Dipole (quarter-wave RPSMA)	5.5 dBi	50 Ω	Fixed/Mob	20 cm.
Pulse RO2408NF	Whip (RPSMA)	8.0 dBi	50 Ω	Fixed/Mob	20 cm.
Linx Technologies ANT-2.4-CW-RAH-RPS	Helical Whip (quarter-wave RPSMA)	1.6 dBi	50 Ω	Fixed/Mob	20 cm.
Linx Technologies ANT-2.4-CW-RH-RPS	Helical Whip (quarter-wave RPSMA)	-0.9 dBi	50 Ω	Fixed/Mob	20 cm.

11.2.2 Wi-Fi Antennas

Part Number	Type	Gain	Impedance	Application	Min. Separation
Pulse W1030	Dipole (quarter-wave RPSMA)	2.0 dBi	50 Ω	Fixed/Mobile	20 cm.

For more information on approved antennas, please consult the manufacturer's website.

Warning

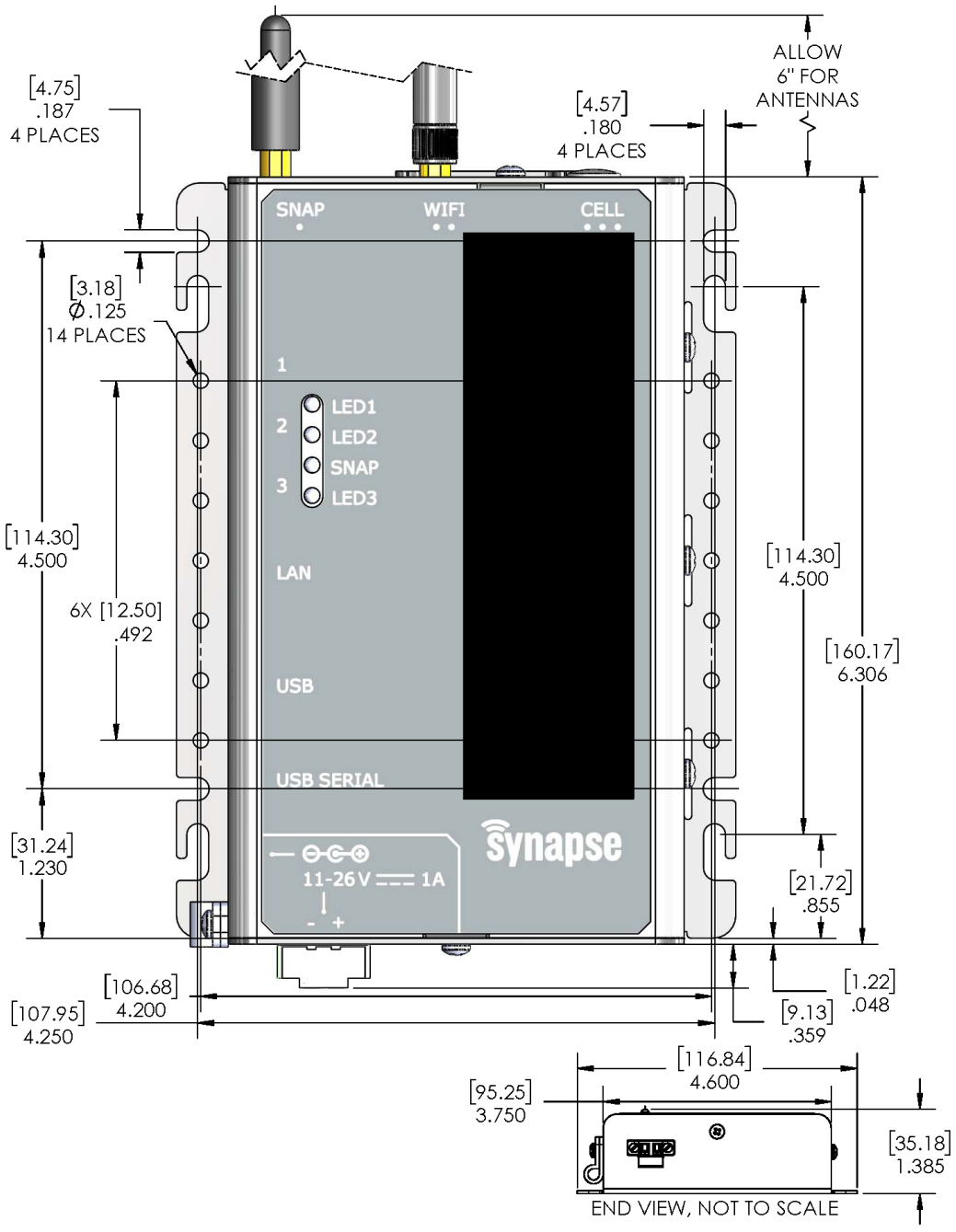
RF Exposure. This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20 cm between

the radiator and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Note

Antenna and transmitters may be co-located or operated in conjunction with this device only if the transmitters do not simultaneously transmit. Otherwise, additional regulatory requirements will apply.

11.3 E20 Dimensions



CHAPTER

TWELVE

TROUBLESHOOTING

12.1 The Ethernet port does not work or eth0 does not appear in ifconfig

Most likely, your MAC address has not been set. If you run `ifconfig -a` and see output similar to:

```
eth_badmac Link encap:Ethernet HWaddr 00:1c:2c:ff:ff:ff
```

Your MAC address has reverted to the default and needs to be set. See SNAPconnect is not working below.

12.2 SNAPconnect is not working

Check for one of the following possible issues:

- SNAPconnect may be out of date.
  Prior versions of SNAPconnect were incompatible with some version of the Python Tornado package or may have failed to obtain the MAC address correctly. Upgrade to the latest version of SNAPconnect by using the instructions in E20-Specific Software Packages.

- Your MAC address is not set.
  From the linux command line, run:

  ```
  sudo fw_setenv ethaddr "00:1c:2c:xx:xx:xx"
  ```

  (Replace the XX entries with your last three octets of your Ethernet MAC address, which should be found on the label on your E20's case. Be sure to use the Ethernet MAC address (labeled ETH MAC) and not the SNAP MAC address or Wi-Fi MAC address.)
  Then reboot.

12.3 Cannot SSH into my E20

You cannot SSH into the E20 as root, or any user account which does not have the password set. Be sure to have set a password for the account you want to use to connect.

12.4 The E20 is slow to boot because it's waiting for the network

Ubuntu will wait up to two minutes during boot up to ensure that every network device configured to come up automatically in DHCP mode acquires an IP address before continuing. The DHCP client will continue to attempt to acquire an address for all DHCP devices periodically in the background. By default, it will spend 60 seconds trying and then wait five minutes before trying again (the timeout and retry times, respectively).

If either your Ethernet or your Wi-Fi is configured to get an IP address automatically in DHCP client mode, and either network is unavailable (or network is congested, or with a slow-to-respond DHCP server), you can have a blocking delay of up to two minutes while the operating system attempts to get IP addresses assigned for all its interfaces. This two-minute period covers both the Ethernet and Wi-Fi interfaces, whether one or both of them are unable to make a network connection.

There are a few ways to address this issue:

1. If you don't care about Ethernet at all, edit `/etc/network/interfaces`, and change the lines:

```
auto eth0
allow-hotplug eth0
```

to:

```
#auto eth0
#allow-hotplug eth0
```

This will disable bringing up the Ethernet. To disable Wi-Fi, just comment the line `auto wlan0` if it isn't already.

2. If you want your network devices brought up, but can tolerate continuing booting potentially before each has an IP address, you can reconfigure the length of the delay. By default, Ubuntu waits first 20 seconds, then 40, then finally 60 seconds before abandoning waiting on all devices to acquire an IP address, for the two-minute total mentioned earlier. You can change these times by calling the dhclient timeout utility::

```
sudo E20-dhclient-setargs --boot-time 10 15 20
```

This should be reflected in the next reboot.

Additionally, you can reduce the DHCP client retry and timeout times:

```
sudo E20-dhclient-setargs --timeout 30 --retry 180
```

After changing this setting, the DHCP client will try for 30 seconds before going to sleep for three minutes and trying again.

If you have an exceptionally slow DHCP server and can tolerate booting before the device receives its IP address, you can set your boot time arguments low and then set a long timeout value with whatever retry parameter suits your needs. It is important that your timeout parameter is long enough to catch the response from your DHCP server before it goes to sleep.

CHAPTER

THIRTEEN

REGULATORY INFORMATION AND CERTIFICATIONS

13.1 RF Exposure Statement

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20cm between the radiator and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

13.2 FCC Certifications and Regulatory Information (USA Only)

13.2.1 FCC Part 15 Class B

These devices comply with part 15 of the FCC rules. Operation is subject to the following two conditions: (1) These devices may not cause harmful interference, and (2) These devices must accept any interference received, including interference that may cause harmful operation.

13.2.2 Radio Frequency Interference (RFI) (FCC 15.105)

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However,there is no guarantee that the interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Reorient or relocate the receiving antenna. Increase the separation between the equipment and receiver. Connect the equipment into an outlet on a circuit different from that of the receiver. Consult the dealer or an experienced radio/TV technician for help. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

13.2.3 Labeling Requirements (FCC 15.19)

This device complies with Part 15 of FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. If the FCC ID for the module inside this product enclosure is not visible when installed inside another device, then the outside of the device into which this product is installed must also display a label referring to the enclosed module FCC ID.

13.2.4 Modifications (FCC 15.21)

Changes or modifications to this equipment not expressly approved by Synapse Wireless, Inc. may void the user's authority to operate this equipment.

13.2.5 Declaration of Conformity

(In accordance with FCC 96-208 and 95-19)

Manufacturer's Name: Synapse Wireless, Inc.

Headquarters: 6723 Odyssey Drive, Huntsville, AL 35806

Synapse Wireless, Inc. declares that the product:

Product Name: E20-0

to which this declaration relates, meet the requirements specified by the
Federal Communications Commission as detailed in the following specifications:

- Part 15, Subpart B, for Class B equipment
- FCC 96-208 as it applies to Class B personal computers and peripherals

The products listed above have been tested at an External Test Laboratory certified per FCC rules and has been
found to meet the FCC, Part 15, Emission Limits. Documentation is on file and available from Synapse Wireless, Inc

13.3 Industry Canada (IC) Statement

This Class B digital apparatus complies with Canadian ICES-003. Cet appareil numérique de la classe B est conforme à la norme NMB-003 du Canada.

13.4 CE Transmit Power

By default, the **SNAPconnect E20** ships with transmit levels that are not within the power levels allowed by CE. It is up to the customer to ensure that their system is CE compliant.

In order to set the SNAP radio's transmit power to CE levels, the following command should be issued from the **SNAPconnect E20**:

```
snap node bridge nvparam 11 0x0200
```