# nRF Sniffer for 802.15.4

**v0.7.2**

**User Guide**

4393_281 / 2021-05-10

# Contents

# Revision history

| Date | Description |
|---|---|
| 2022-06-24 | Added note about the nRF USB port to Setting up hardware for nRF Sniffer on page 16 |
| 2021-05-10 | First release |

# 1 Introduction

The nRF Sniffer for 802.15.4 is a tool for learning about and debugging applications that are using protocols based on IEEE 802.15.4, such as Thread and Zigbee. It provides a near real-time display of 802.15.4 packets that are sent back and forth between devices, even when the link is encrypted.

When developing a 802.15.4-compatible product, knowing what happens over-the-air between devices can help you identify and fix issues quickly.

The nRF Sniffer captures packets transmitted by nearby devices on a selected radio channel. You can start the capture manually from Wireshark, a free software tool that captures wireless traffic and reproduces it in a readable format, or using a Python script. Use either of these methods to create packet capture files, from which you can extract data in Wireshark. This data can include destination and source addresses, personal area network identifiers, and packet payloads.

The nRF Sniffer for 802.15.4 comes with an extcap plugin for capturing packets in Wireshark. This plugin can also be installed as a Python module for use in a script.

Wireshark is also able to analyze data exchanged over higher-level protocols, such as Thread and Zigbee. You can configure the Sniffer to report out-of-band metadata, such as channel, received signal strength indicator (RSSI), and link quality inidcator (LQI).

**Supported devices**

- nRF52840 Development Kit (PCA10056)
- nRF52840 Dongle (PCA10059)

**Supported operating systems**

- Windows 10
- 64-bit macOS 10.14 or later
- Ubuntu Linux (check the Wireshark prerequisites for version compatibility)

# 2 Installing nRF Sniffer for 802.15.4

The nRF Sniffer for 802.15.4 consists of firmware that is programmed onto a development kit or dongle and a capture plugin for Wireshark that records and analyzes the detected data.

Before you set up the nRF Sniffer, make sure that you have satisfied the following prerequisites:

- Install Python v3.7 or later on your computer if you do not already have it.
- Clone the nRF Sniffer for 802.15.4 GitHub respository into a folder of your choice. In the following sections, this folder will be referred to as *Sniffer_Software*.
- Program the firmware to the development kit or dongle.
- Install Wireshark.
- Install the nRF Sniffer capture plugin.

See the following sections for details about programming the firmware and installing Wireshark and the capture plugin.

## 2.1 Programming the nRF Sniffer firmware

You must connect a development kit or dongle running the nRF Sniffer firmware to your computer to use the nRF Sniffer for 802.15.4.

See Supported development kits and dongles for a list of development kits and dongles that can run the nRF Sniffer firmware.

There are various ways to program the nRF Sniffer firmware. The following instructions use nRF Connect Programmer, but you can also use the command line tool **`nrfjprog`** (which is part of the nRF Command Line Tools).

To program your development kit or dongle, complete the following steps:

1. Install nRF Connect Programmer.

   See Installing the Programmer for instructions.
2. On macOS and Linux, install the SEGGER J-Link Software.

   > **Note:** On Windows, the J-Link software is included in nRF Connect for Desktop, so you can skip this step.
3. Locate the firmware hexadecimal file for your development kit or dongle.

   All firmware hexadecimal files are located in `Sniffer_Software/nrf802154_sniffer/`. Use the suitable file for your development kit or dongle:

| Development kit/dongle | Firmware file name |
|---|---|
| nRF52840 DK (PCA10056) | `nrf802154_sniffer.hex` |
| nRF52840 Dongle (PCA10059) | `nrf802154_sniffer_dongle.hex` |

*Table 1: Firmware file names*

4. Follow the instructions in Programming a Development Kit or the nRF51 Dongle or Programming the nRF52840 Dongle to program the HEX file.

## 2.2 Installing Wireshark on Windows and macOS

The Wireshark installation procedure for Windows and macOS is identical.

To install Wireshark on Windows or macOS:

1. Go to the Wireshark download page website.
2. From the **Stable Release** list in the **Download Wireshark** section, click the release package for your operating system.

   The download starts automatically.
3. Install the tool.

Then install the nRF Sniffer capture plugin, as described in Installing the nRF Sniffer capture plugin in Wireshark on page 17.

## 2.3 Installing Wireshark on Ubuntu Linux

Installing Wireshark on Ubuntu Linux requires creating a custom user group.

To install Wireshark on Ubuntu Linux:

1. Download Wireshark for Ubuntu Linux:
   a) Go to Wireshark download page.
   b) From the table at the bottom of the page, download the Wireshark standard package or the latest stable PPA for Ubuntu Linux distribution.
   c) Install the package on your system.
   d) During the download and installation process, make sure to create the *wireshark* user group and allow all members of that group to capture packets.
2. Add the correct *user_name* to the *dialout* user group by typing `sudo usermod -a -G dialout user_name`.
3. Log out and log in again to apply the new user group settings.

Then install the nRF Sniffer capture plugin, as described in the following section.

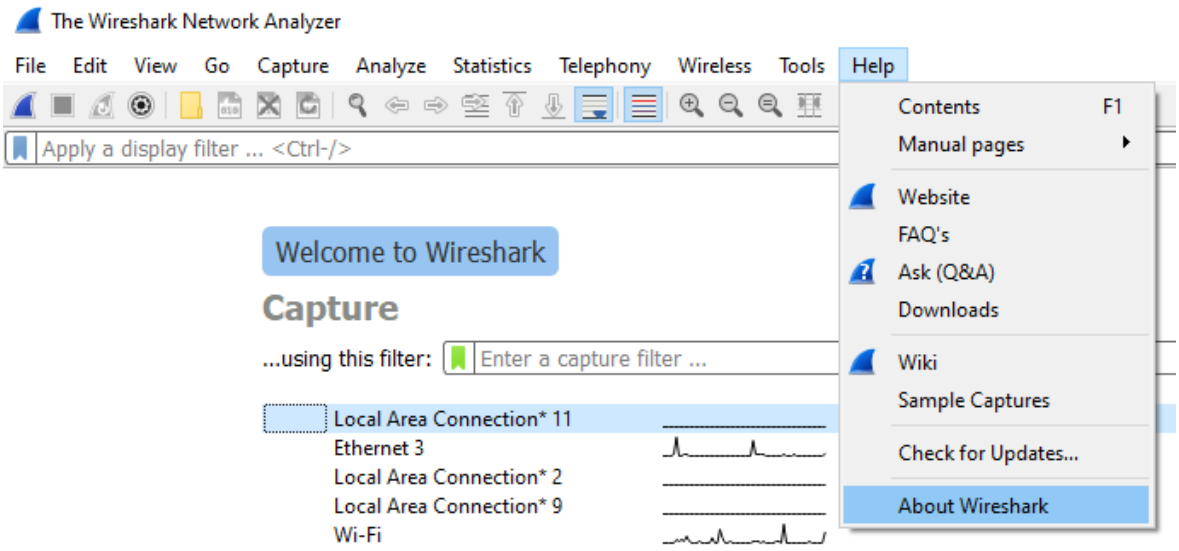
## 2.4 Installing out-of-band metadata Lua plugin

If you are using a Wireshark version earlier than v3.0 and you want to inspect the out-of-band metadata such as RSSI, install the Lua dissector plugin that is available in the nRF Sniffer for 802.15.4 package.

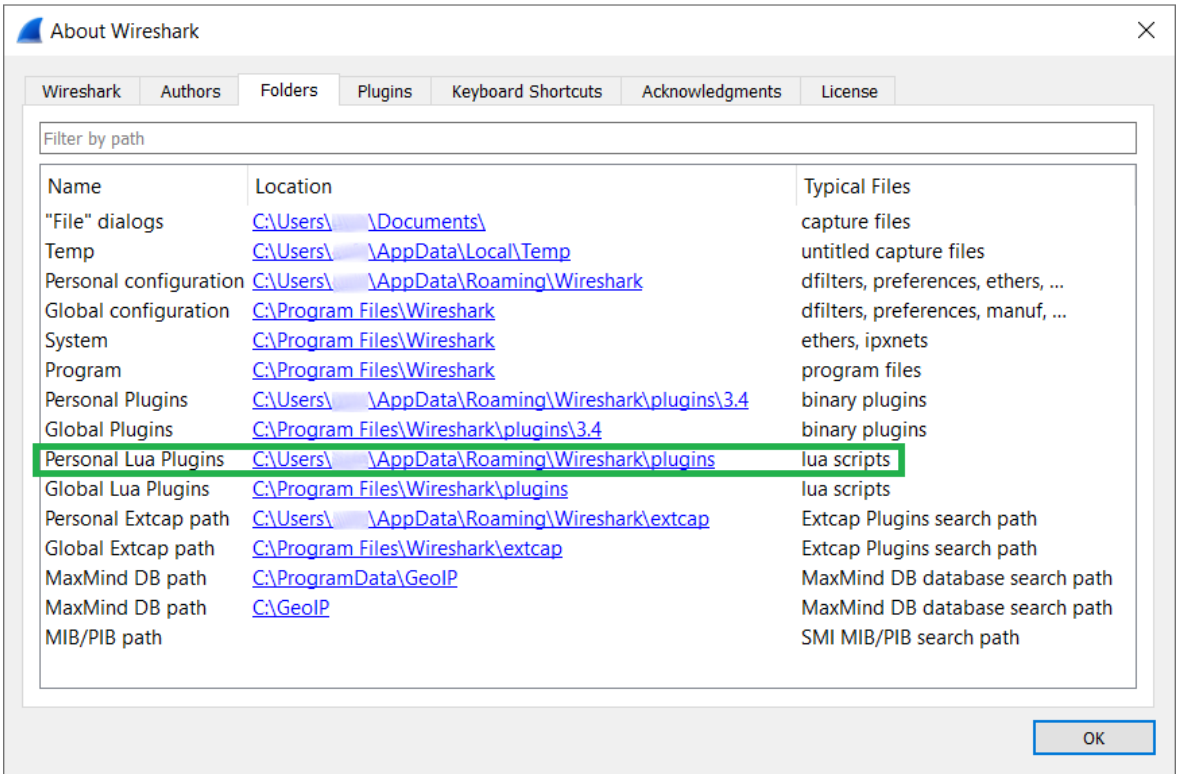
If you are using Wireshark v3.0 or later, installing the custom Lua dissector is not required to capture the additional out-of-band metadata. In such case, use the **IEEE 802.15.4 TAP** option, as described in Running nRF Sniffer in Wireshark with custom options on page 20.

To install the custom Lua dissector plugin:

1. Open Wireshark.
2. Go to **Help** > **About Wireshark** (on Windows or Linux) or **Wireshark** > **About Wireshark** (on macOS).

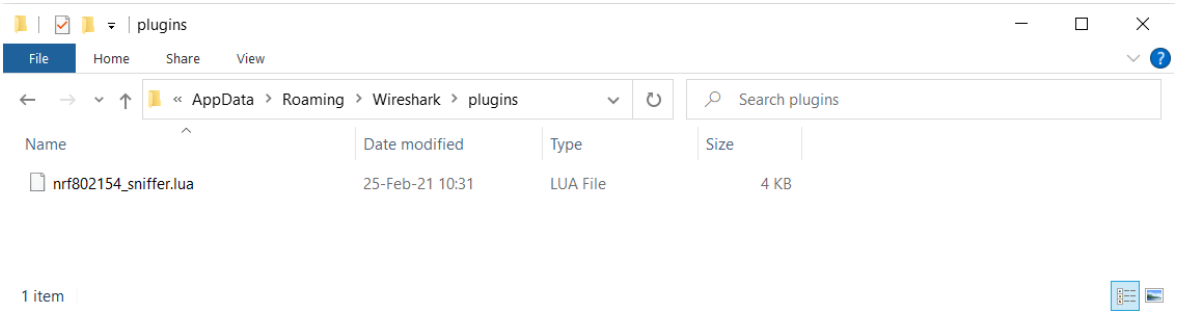



3. Select the **Folders** tab.
4. Double-click the location for the **Personal Lua Plugins** to open this folder.




> **Note:** If the folder does not open, go to the `\AppData\Roaming\Wireshark\` folder and create the `plugins` folder.

5. Copy the `nrf802154_sniffer.lua` file from the *`Sniffer_Software/`* `nrf802154_sniffer/` folder into this folder.




You can now select the Lua dissector in the out-of-band metadata settings, as described in Running nRF Sniffer in Wireshark with custom options on page 20.

# 3 Configuring Wireshark for nRF Sniffer for 802.15.4

The nRF Sniffer for 802.15.4 must be configured for capturing and analyzing packets exchanged on Thread and Zigbee networks.

## 3.1 Configuring Wireshark for Thread

Capturing packets on a Thread network requires configuring at least the IEEE 802.15.4 decryption keys. Additionally, you can also configure the CoAP port and the 6loWPAN settings.

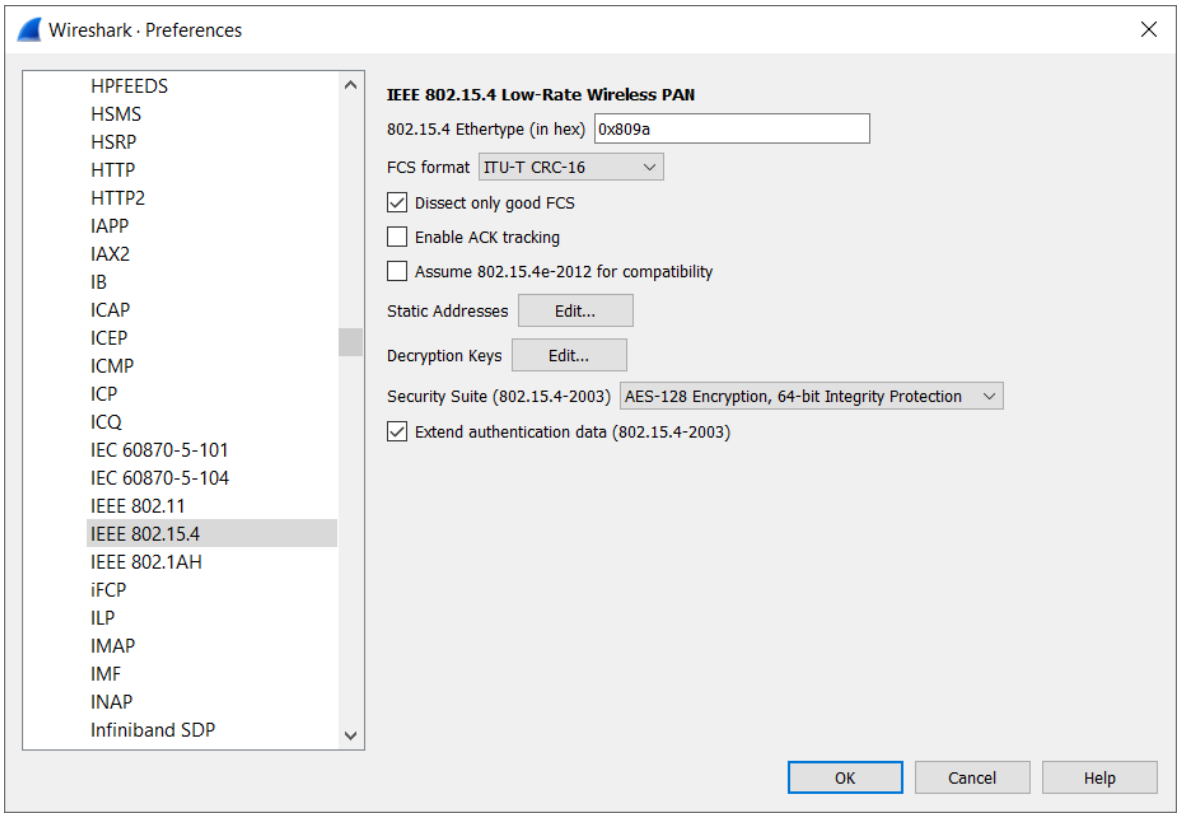
### 3.1.1 Configuring decryption keys for Thread

You must configure IEEE 802.15.4 decryption keys to decode packets exchanged on the network and display the data in a readable format.

You need to know the Thread decryption key before you start configuring it in Wireshark. For example, if one of the devices in the Thread network has the OpenThread CLI enabled, you can check the decryption key by calling the `masterkey` CLI command.

To configure the decryption keys:

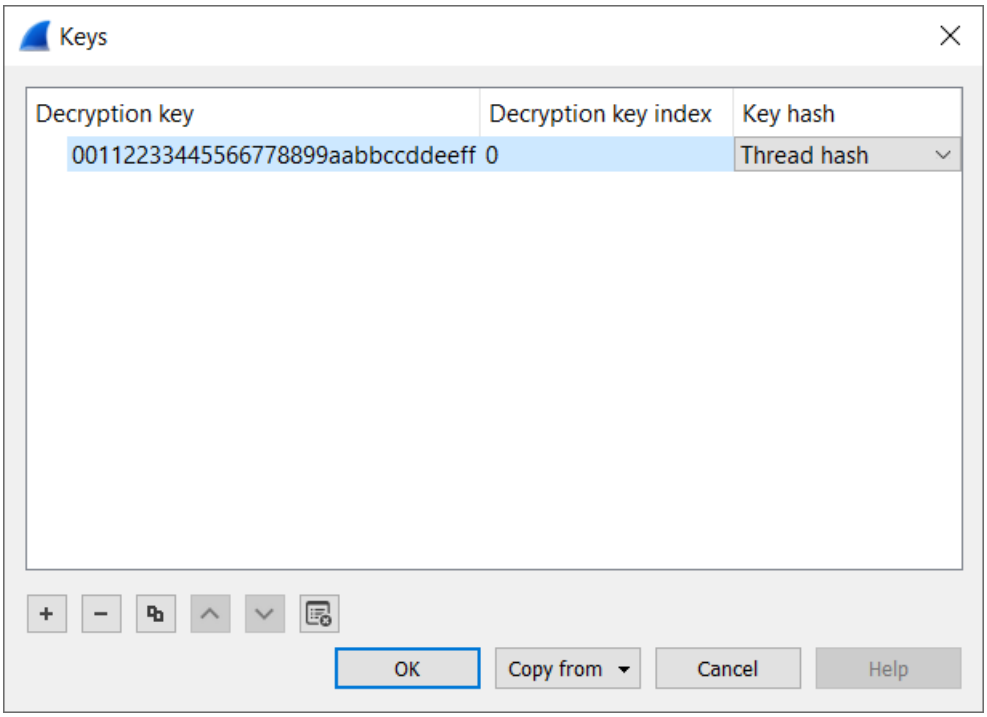
1. In Wireshark, go to **Edit** > **Preferences...**.
2. In the **Preferences** section list, go to **Protocols** > **IEEE 802.15.4**.




3. Click the **Edit...** button next to **Decryption Keys**.
4. In the **Keys** window:

a) Click **+** and add the **Decryption key** value with **Decryption key index** set to `0` and **Key hash** set to `Thread hash`.

For example, for the Thread network that includes devices based on Thread examples from nRF5 SDK for Thread and Zigbee v4.2.0, set the decryption key value to `00112233445566778899aabbccddeeff`.




b) Click **OK** to close the window.

5. Click **OK** to save the decryption keys for Thread.

Now you can start capturing data from the Thread network and display the information in a readable format.

## 3.1.2 Configuring CoAP port for Thread

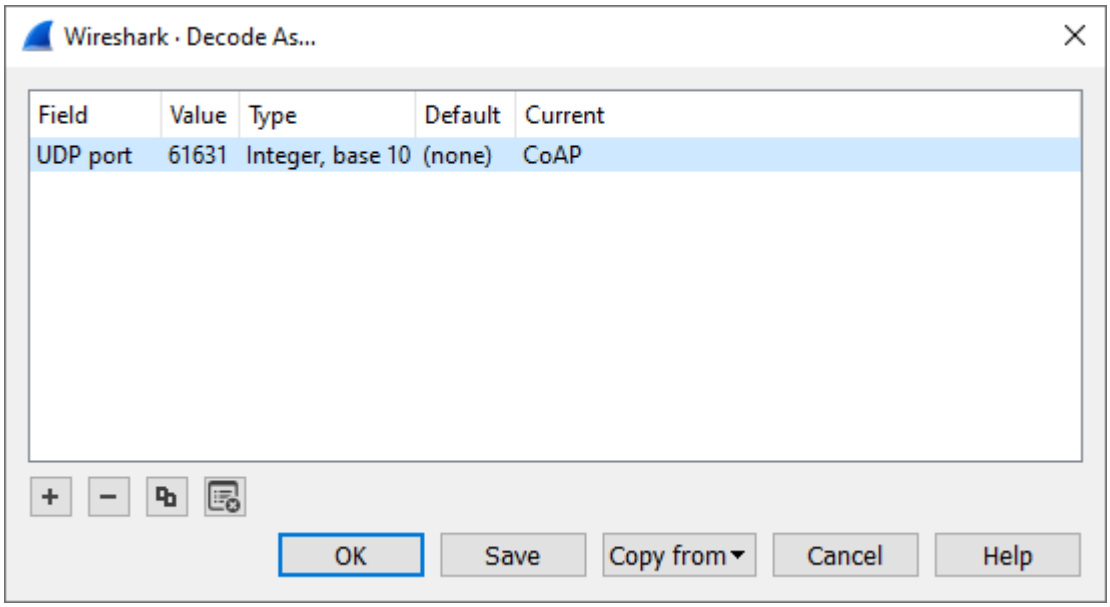
The Thread network uses the CoAP protocol on port *61631* for network management. You must configure this port in Wireshark if you want to correctly decode network management packets sent over this port.

### 3.1.2.1 Configuring CoAP port using Decode As

You can apply the CoAP protocol settings once on a per-capture basis using the **Decode As** option.

To configure the CoAP port using this option:

1. In Wireshark, go to **Analyze** > **Decode As...**.
2. In the **Decode As...** settings window, click the **+** button to add a new entry with the **Field** set to **UDP port**, value set to `61631`, and **Current** set to **CoAP**.



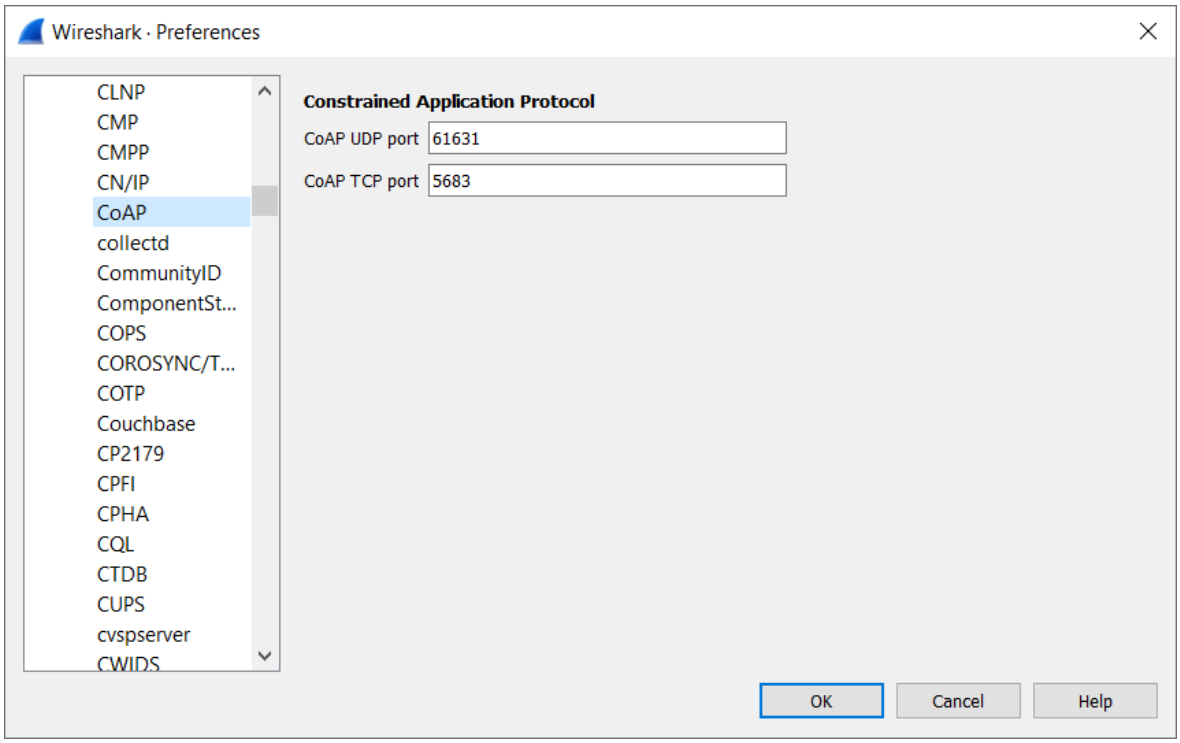

3. Click **OK** to save the **Decode As...** settings.

### 3.1.2.2 Configuring CoAP port using Preferences

You can apply the CoAP protocol settings globally by defining the CoAP port number in Wireshark Preferences.

To configure the CoAP port using this option:

1. In Wireshark, go to **Edit** > **Preferences...**.
2. In the **Preferences** section list, go to **Protocols** > **CoAP**.




3. Set the **CoAP UDP port** to `61631`.
4. Click **OK** to save the CoAP port settings.

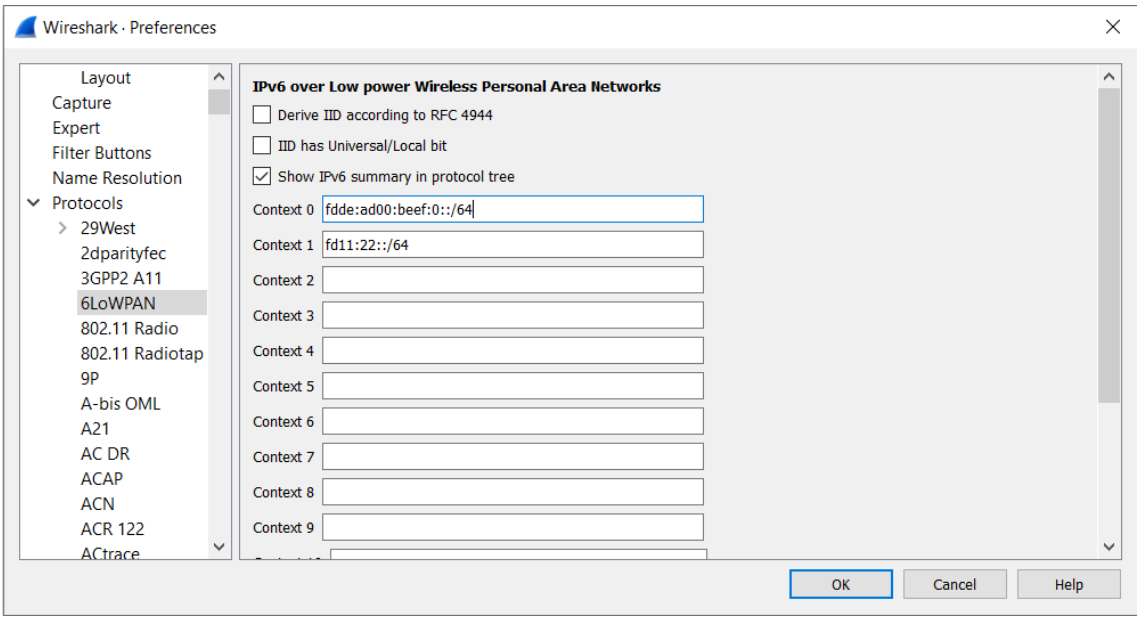
## 3.1.3 Configuring 6loWPAN context

6loWPAN defines contexts that are used to shorten IPv6 addresses sent over-the-air. Configuring the 6loWPAN context ensures that the correct IPv6 address is displayed during packet analysis.

You can configure different 6loWPAN contexts depending on the Thread Network Data.

To configure the 6loWPAN contexts used by Thread examples:

1. In Wireshark, go to **Edit** > **Preferences...**.
2. In the **Preferences** section list, go to **Protocols** > **6loWPAN**.
3. Set the following contexts to the provided values:
   a) In the **Context 0** field, add `fdde:ad00:beef:0::/64`.
   b) In the **Context 1** field, add `fd11:22::/64`.




4. Click **OK** to save the 6loWPAN contexts for Thread.

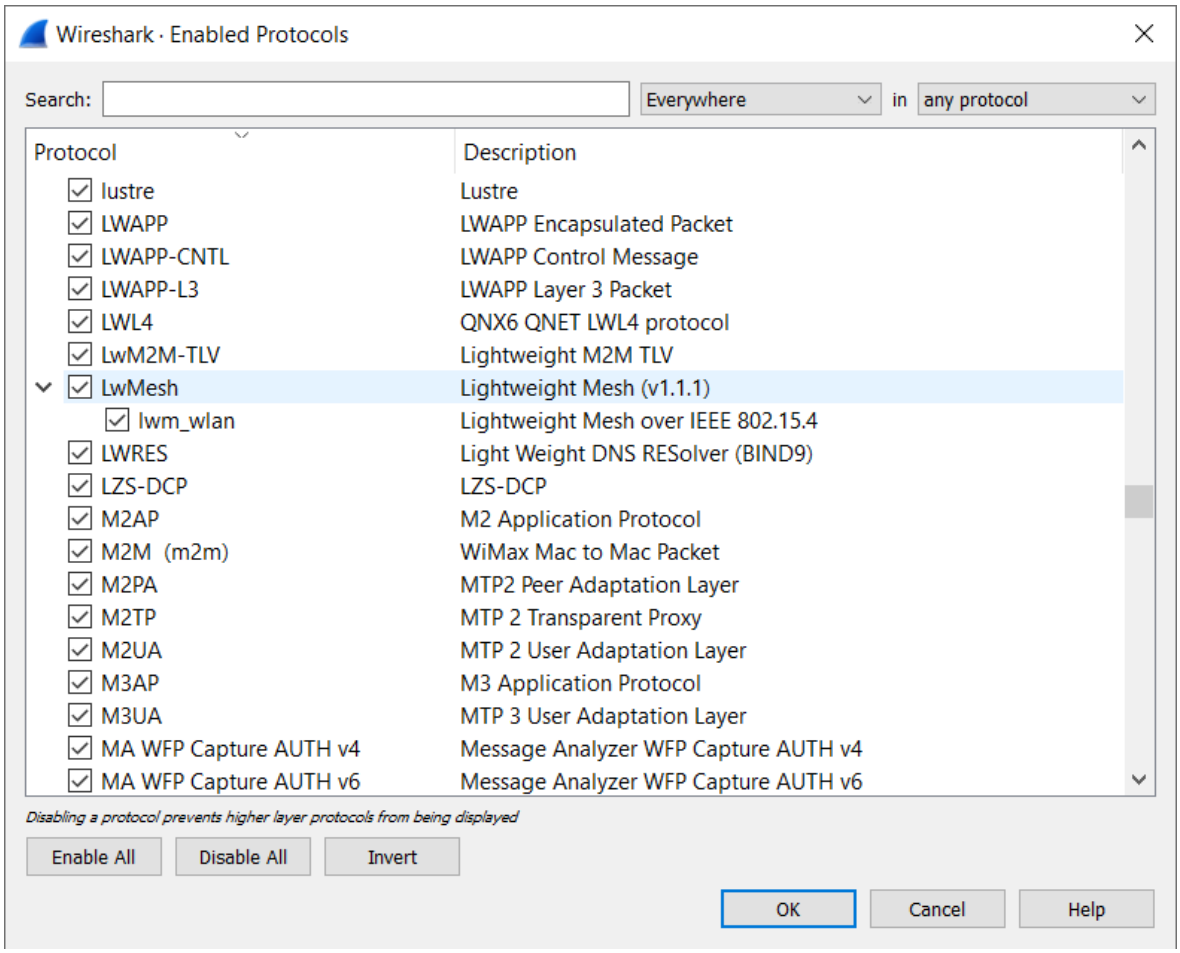
## 3.1.4 Disabling unwanted protocols

If Wireshark uses incorrect dissectors to decode a Thread message, you can optionally disable unwanted protocols.

To disable unwanted protocols when capturing data from a Thread network:

1. In Wireshark, go to **Analyze** > **Enabled Protocols...**.
2. In the **Enabled Protocols** window, disable unwanted protocols by unchecking the field next to their name.

   For example, you might want to disable **LwMesh**, **ZigBee**, and **ZigBee Green Power**.




**3.** Click **OK** to save the settings.

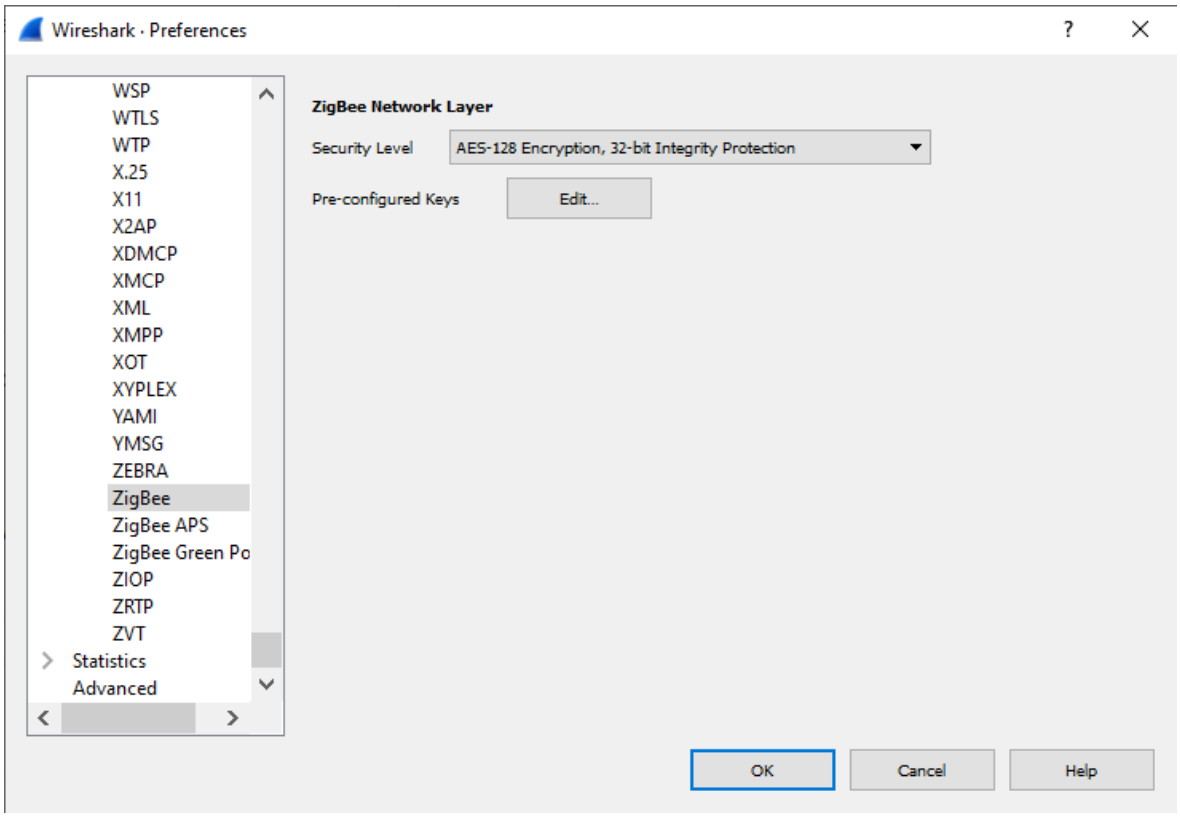
## 3.2 Configuring Wireshark for Zigbee

Additional Wireshark configuration is required to start capturing data from Zigbee samples in the nRF Connect SDK or from Zigbee examples in the nRF5 SDK for Thread and Zigbee.

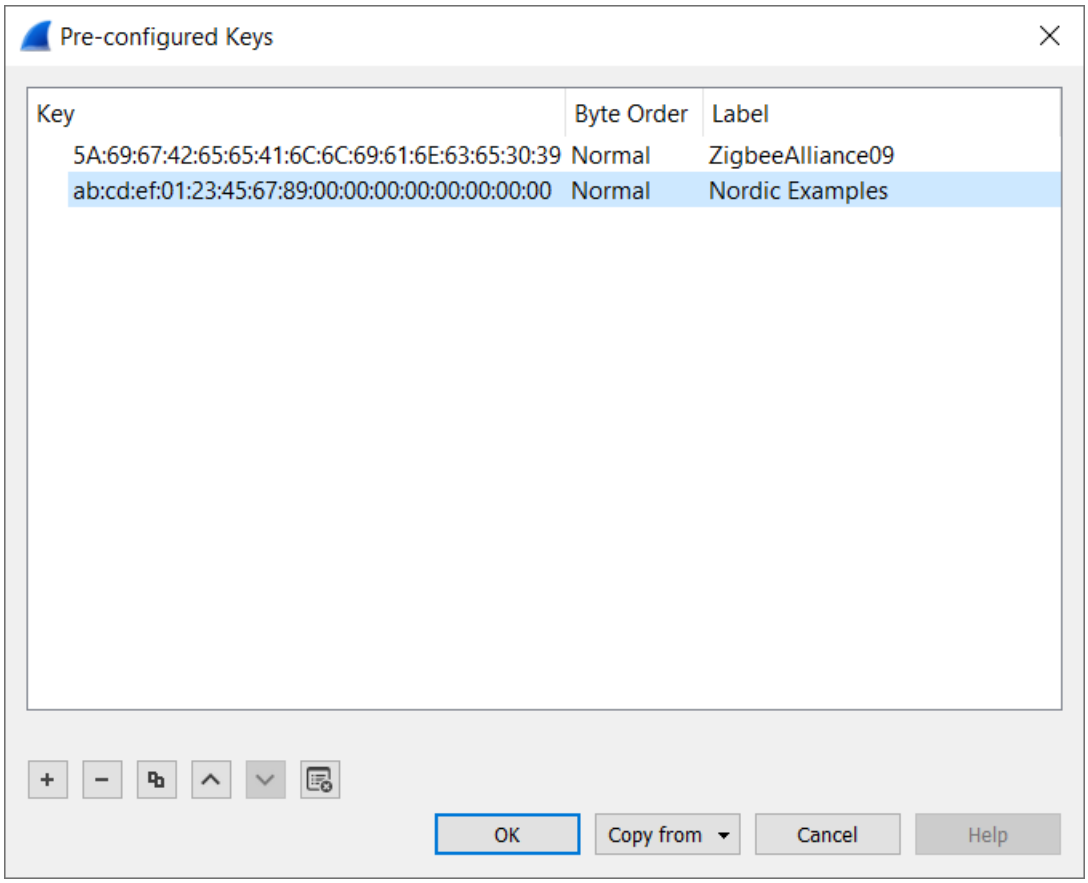
To capture data from Zigbee examples and samples:

**1.** In Wireshark, go to **Edit** > **Preferences...**.

**2.** In the **Preferences** section list, go to **Protocols** > **ZigBee**.




3. Click the **Edit...** button to add the preconfigured keys.
4. In the **Pre-configured Keys** window:
   a) Click **+** and add the key `5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39` with **Byte Order** set to **Normal** and **Label** set to `ZigbeeAlliance09`.
   b) Click **+** and add the key `ab:cd:ef:01:23:45:67:89:00:00:00:00:00:00:00:00` with **Byte Order** set to **Normal** and **Label** set to `Nordic Examples`.




c) Click **OK** to close the window.

**5.** Click **OK** to save the preferences for Zigbee.

# 4 Capturing data with the nRF Sniffer

You can start capturing manually from Wireshark or using a Python script.

The nRF Sniffer for 802.15.4 listens on the specified channel to pick up as many packets as possible from as many devices as possible. The default channel is *11*. To listen on a different channel, you can either run nRF Sniffer with custom options (if you are starting the capturing process manually) or set the specific channel when integrating the nRF Sniffer module into your script (if you are running the capture from a script).

## 4.1 Setting up hardware for nRF Sniffer

Before you start sniffing, place the development kit or dongle that runs the nRF Sniffer for 802.15.4 firmware near the devices that are communicating. The hardware setup is the same for all supported methods of capturing data, whether Wireshark or a custom Python script.

Connect the nRF Sniffer development kit or dongle to your computer and turn it on. Then place it next to the devices that you want to sniff.

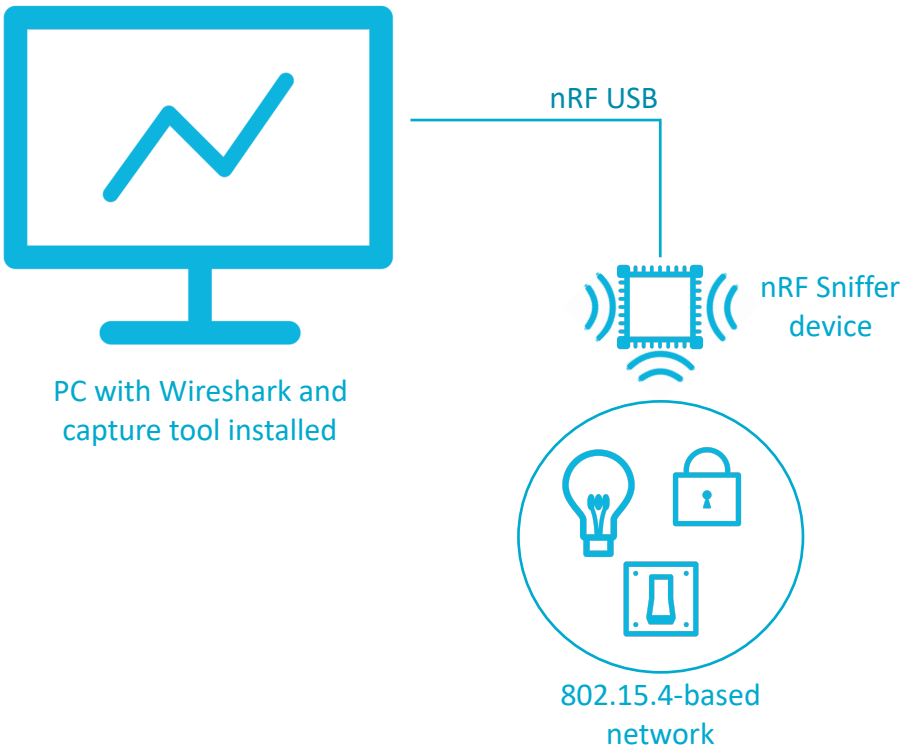



*Figure 1: Hardware setup*

**Note:** Make sure to use the nRF USB (**J3**) port to connect the nRF Sniffer device with the PC. For example, see the following image of the nRF52840 DK, where the nRF USB port is located next to the IF BOOT/RESET (**SW5**) button.

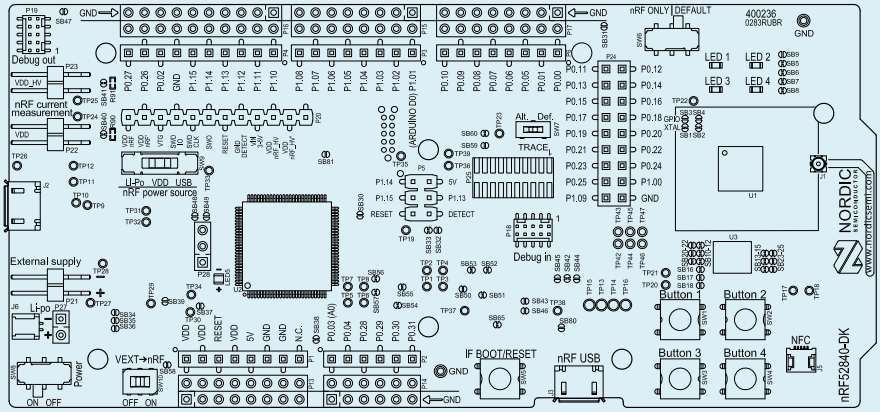



*Figure 2: nRF52840 DK (PCA10056) front view*

The firmware of the nRF Sniffer for 802.15.4 exposes the USB interface with the following VID and PID values:

```
NORDICSEMI_VID = 0x1915
SNIFFER_802154_PID = 0x154A
```

You can use these values to identify the nRF Sniffer interface in your network setup.

# 4.2 Capturing data in Wireshark

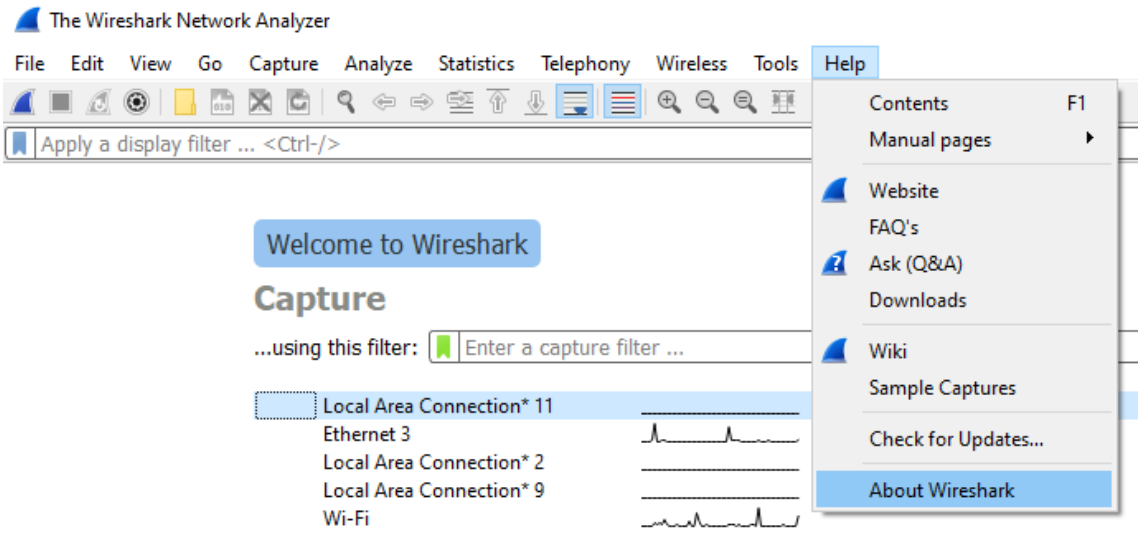
You can start capture manually from Wireshark with or without the out-of-band metadata. Capturing in Wireshark requires installing an nRF Sniffer plugin.

## 4.2.1 Installing the nRF Sniffer capture plugin in Wireshark

The nRF Sniffer for 802.15.4 software sends commands to the nRF Sniffer hardware through the serial port and reads the captured frames. The software can be installed as an external capture plugin in Wireshark. You need to install the plugin only if you plan to use the nRF Sniffer as a Wireshark capture interface.

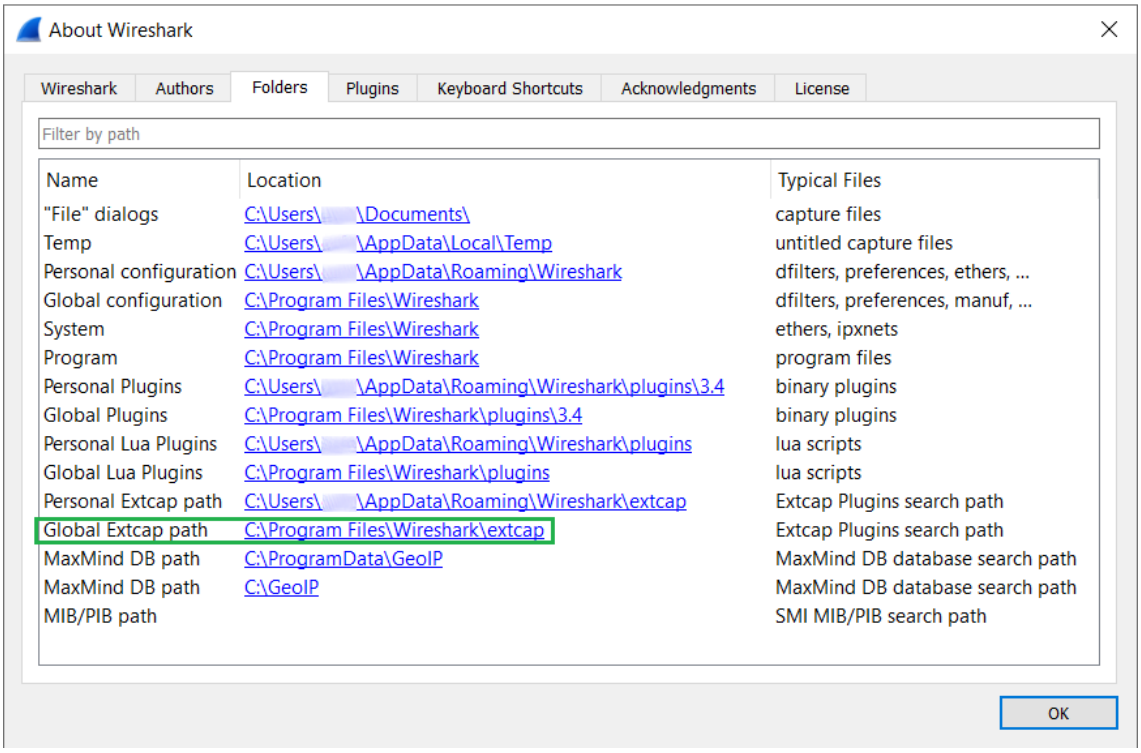
To install the nRF Sniffer capture plugin, complete the following steps:

1. Install the pySerial module:
   a) Open a command window.
   b) Install the pySerial module by typing `pip install pyserial` (on Windows) or `sudo pip install pyserial` (on Linux or macOS).
   c) Close the command window.
2. Copy the Sniffer capture plugin into Wireshark's folder for external capture plugins:
   a) Open Wireshark.
   b) Go to **Help** > **About Wireshark** (on Windows or Linux) or **Wireshark** > **About Wireshark** (on macOS).



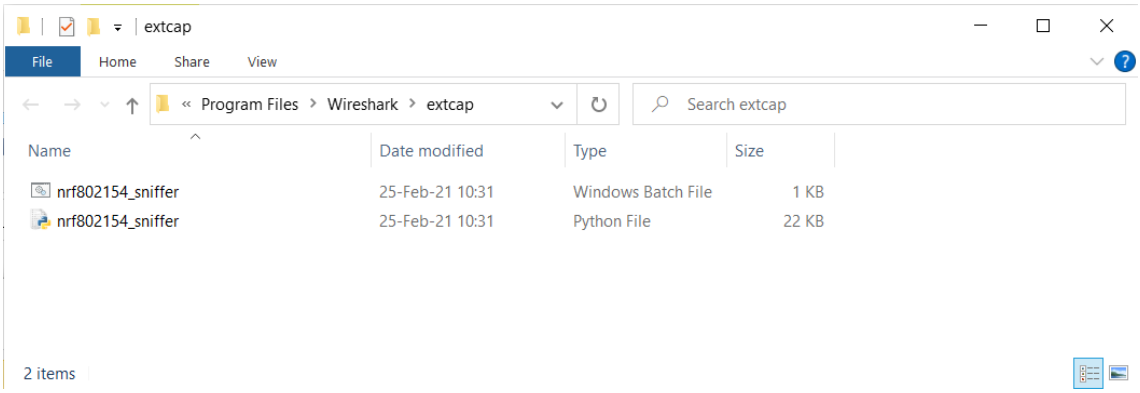

c) Select the **Folders** tab.

d) Open the plugin folder by double-clicking the location for the **Global Extcap path**. This folder contains plugins available for all users and is available for Wireshark versions earlier than v3.0. Plugins copied into the **Personal Extcap path** folder are installed only for one user, and this folder is not available in the older versions of Wireshark.




e) Copy the following files from the *`Sniffer_Software`*`/nrf802154_sniffer/` folder into this folder:

- On all operating systems: `nrf802154_sniffer.py`
- Additionally on Windows: `nrf802154_sniffer.bat`



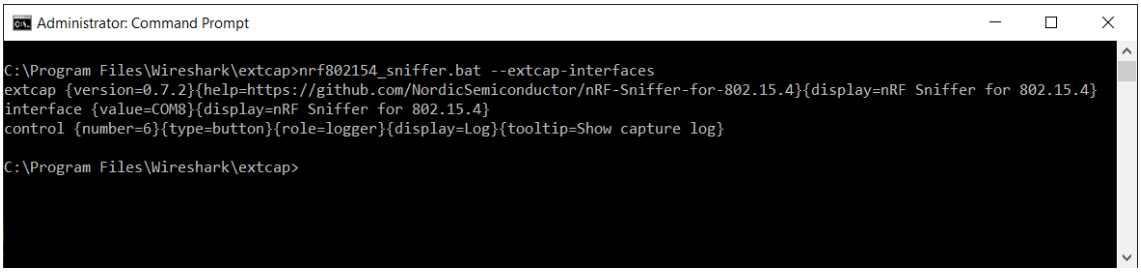

**3.** Make sure that the nRF Sniffer files run correctly:

a) Open a command window in Wireshark's folder for external capture plugins.

b) Run the Sniffer tool to list available interfaces.

On Windows, type `nrf802154_sniffer.bat --extcap-interfaces`. On macOS or Linux, type `nrf802154_sniffer.py --extcap-interfaces`.

You should see a series of strings, similar to what is shown in the following screenshot.




c) If the previous step returned an error, verify that Python 3 is accessible.

On Windows, enter `python --version`. On macOS or Linux, enter `python3`. If the command cannot be found or the version is wrong, make sure that Python v3.7 or later is in your path and that it is the first Python version in the path.

d) For macOS or Linux: Verify that the `nrf802154_sniffer.py` file has the `x` permission.

If the `x` permission is missing, add it using `chmod +x nrf802154_sniffer.py`.

**4.** Refresh the interfaces in Wireshark by selecting **Capture** > **Refresh Interfaces** or pressing **F5**.

You should see that nRF Sniffer for 802.15.4 is displayed as one of the interfaces on the start screen.

## 4.2.2 Running nRF Sniffer in Wireshark

To start sniffing, open Wireshark and start recording packets.

When you open Wireshark, the Wireshark main window appears. It includes the Wireshark hardware interfaces connected to the nRF Sniffer.

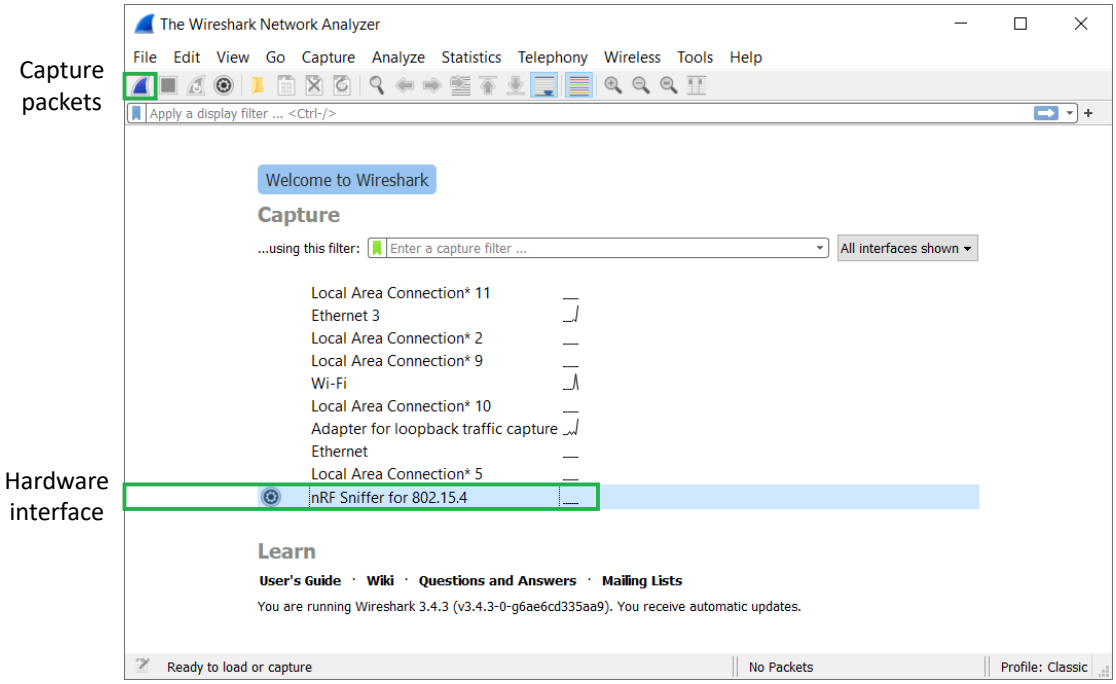



*Figure 3: Wireshark capture screen*

To start sniffing with the default settings, use one of the following options:

- Double-click on the nRF Sniffer for 802.15.4 hardware interface.
- Select the nRF Sniffer for 802.15.4 hardware interface and click the **Capture packets** button.

By default, both these options start the capturing process on channel *11* without capturing the out-of-band metadata. If you want to capture this kind of data or listen on a different channel, run nRF Sniffer with custom options.

Wireshark begins capturing data from the nRF Sniffer for 802.15.4 hardware interface, allowing you to inspect captured data.

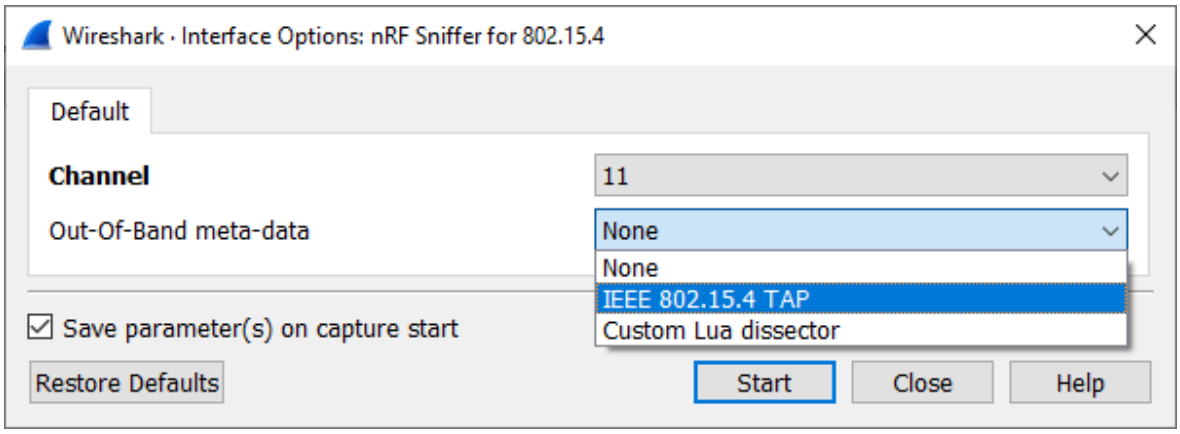
## 4.2.3 Running nRF Sniffer in Wireshark with custom options

To start listening on a custom channel and with custom out-of-band metadata settings, run the capturing tool in Wireshark from the Interface Options window.

If you are using a Wireshark version earlier than v3.0, make sure to install the out-of-band metadata Lua dissector plugin before running the nRF Sniffer with custom out-of-band metadata settings.

To start sniffing:

1. In the main window of Wireshark, click the gear icon next to the **nRF Sniffer for 802.15.4** hardware interface entry to open the **Interface Options** menu.
2. In you want to specify the **Channel** on which the packets are to be captured, use the **Channel** drop-down menu to select it.
3. If you want to get additional out-of-band metadata, use the **Out-Of-Band meta-data** drop-down menu to select the metadata type that corresponds to your version of Wireshark:
   - If you are using Wireshark v3.0 or later, select **IEEE 802.15.4 TAP**.
   - If you are using a Wireshark version earlier than v3.0, select **Custom Lua dissector**.




4. Optionally, you can check the **Save parameter(s) on capture start** option to save the settings for future captures. Keeping this option unchecked will reset the settings to the default values for the next capture.
5. Click **Start** to run nRF Sniffer.

Wireshark begins capturing data from the nRF Sniffer for 802.15.4 hardware interface, allowing you to inspect captured data.

# 4.3 Capturing data using a script

The nRF Sniffer for 802.15.4 can be used in Python scripts to capture packets into a `pcap` file, which you can then open and inspect in Wireshark. Using this option requires installing the capture plugin as a Python module and then integrating it into your script.

## 4.3.1 Installing the nRF Sniffer Python module

You can install the capture tool as a Python module and use this module programmatically in custom Python scripts. The module exposes an API that allows you to start and stop the capture.

To install the nRF Sniffer Python module, complete the following steps:

1. Open the *`Sniffer_Software`* folder.
2. Open a command window in the folder.
3. Install the script by typing the following command:

```
python -m easy_install .
```

Then integrate this module into your custom Python script and use it alongside the nRF Sniffer hardware.

## 4.3.2 Integrating nRF Sniffer Python module into a script

In your script, include the nRF Sniffer Python module and specify the parameters for the API function that starts the capture process. These mandatory and optional parameters define what packets are saved to the `pcap` file.

To integrate the nRF Sniffer Python module into your script:

1. Open your custom Python script and include the nRF Sniffer module:

```
from nrf802154_sniffer import Nrf802154Sniffer
```

2. Check the name of the port to which you connected the nRF Sniffer device. The name is used to set the *dev* parameter.

3. Check the number of the channel on which you want to listen for packets. The number is used to set the *channel* parameter.
4. At the point in your script where you want to start the capture process, add the following lines to start the capture. Use the parameter values from the previous steps. For example:

```
sniffer = Nrf802154Sniffer()
sniffer.extcap_capture(fifo="file.pcap", dev="/dev/ttyACM3", channel=26)
```

In this code, the nRF Sniffer script captures packets from the sniffer on port */dev/ttyACM3* on the channel *26* and saves the results to the `file.pcap` file.

See the following table for the description of all parameters of the `extcap_capture()` function and their possible values.

| Parameter | Type | Description |
|---|---|---|
| file | Mandatory | Defines the name of the `pcap` file to which the captured packets will be saved. The parameter value can also include the path to the file directory if you want to save it in a custom directory. By default, the script saves the file in the working directory. |
| dev | Mandatory | Defines the serial port used to communicate with the nRF Sniffer hardware. |
| channel | Mandatory | Specifies the 802.15.4 radio channel number on which the nRF Sniffer listens for packets. |
| metadata | Optional | Specifies the metadata type for the packet capture. It can have one of the following values:<br>• `None` - No metadata is selected for capture. This is the default setting.<br>• `"ieee802154-tap"` - Selects the IEEE 802.15.4 TAP metadata type for capture. Use this parameter if you are using Wireshark v3.0 or later.<br>• `"user"` - Selects the custom Lua dissector metadata type for capture. Use this parameter if you are using a Wireshark version earlier than v3.0. Make sure to install the OOB metadata Lua dissector plugin in Wireshark before inspecting the captured data if you use this metadata type. |
| control_in | Unused | Specifies a file that Wireshark is going to use to control the capture plugin during run time. Currently unused. |
| control_out | Unused | Specifies a file that Wireshark is going to use to control the capture plugin during run time. Currently unused. |

5. At the point in your script where you want to stop the capture process, add the following lines:

```
sniffer.stop_sig_handler()
```

> **Note:** You can add the lines that start and stop the script multiple times in your script. Make sure to stop the capture before you start a new capture process.

When you run the script with the nRF Sniffer hardware, the nRF Sniffer captures packets and saves the results into the `pcap` file. Open this file in Wireshark to inspect captured data.

# 5 Inspecting captured data

The nRF Sniffer passes all 802.15.4 packets to Wireshark, where they are wrapped in a header containing useful meta-information not present in the packet itself. Wireshark dissects the packets and separates the actual packet from the meta-information.

When you browse captured packets, select a packet in the **packet list** to show the breakdown of that packet in the **packet details pane**. The hexadecimal view of the packet is shown in the **packet bytes pane**. Click a value in the details to highlight it among the bytes, or click on the bytes to highlight it in the details.

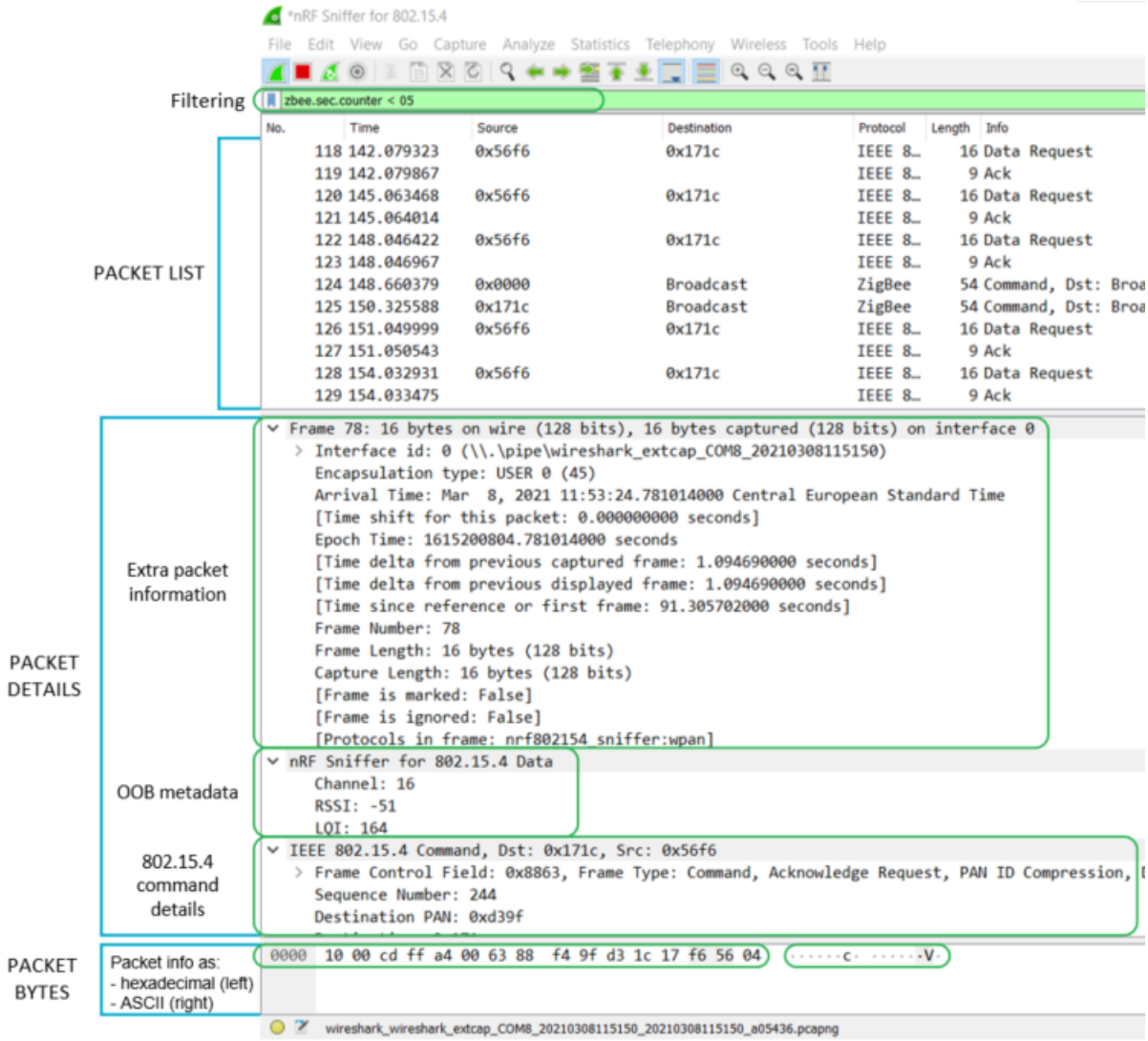
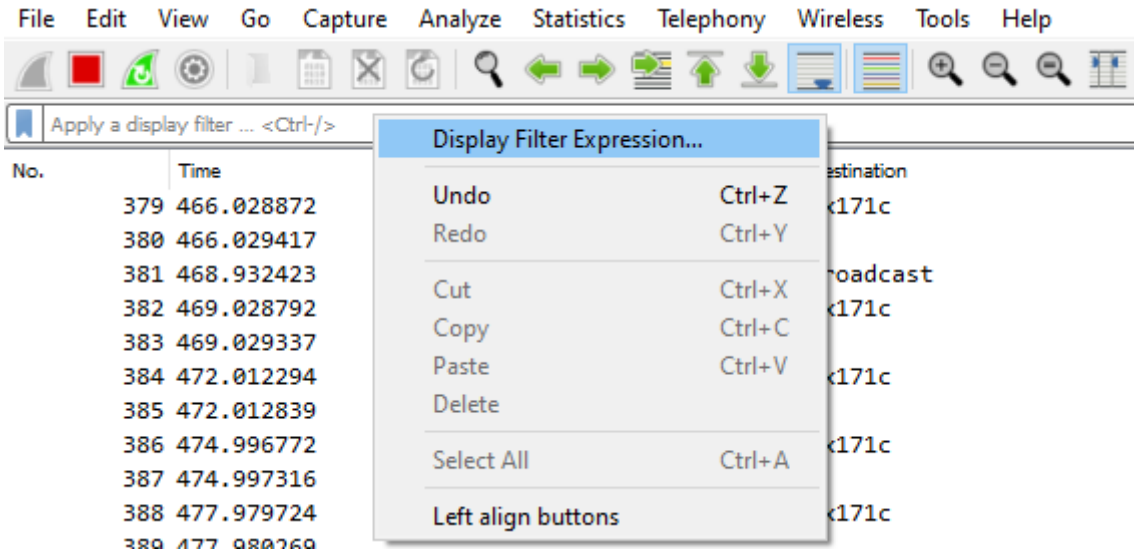



*Figure 4: Wireshark interface*

Use display filters to display a chosen packet subset. To open the filter menu and construct a filter:

1. Right-click the filtering bar.
2. Click **Display Filter Expression...**.




Most filters are based on the values of the packets, such as length or access address. The filter expressions use Boolean operators (`&&` `||` `==` `!=` `!`).

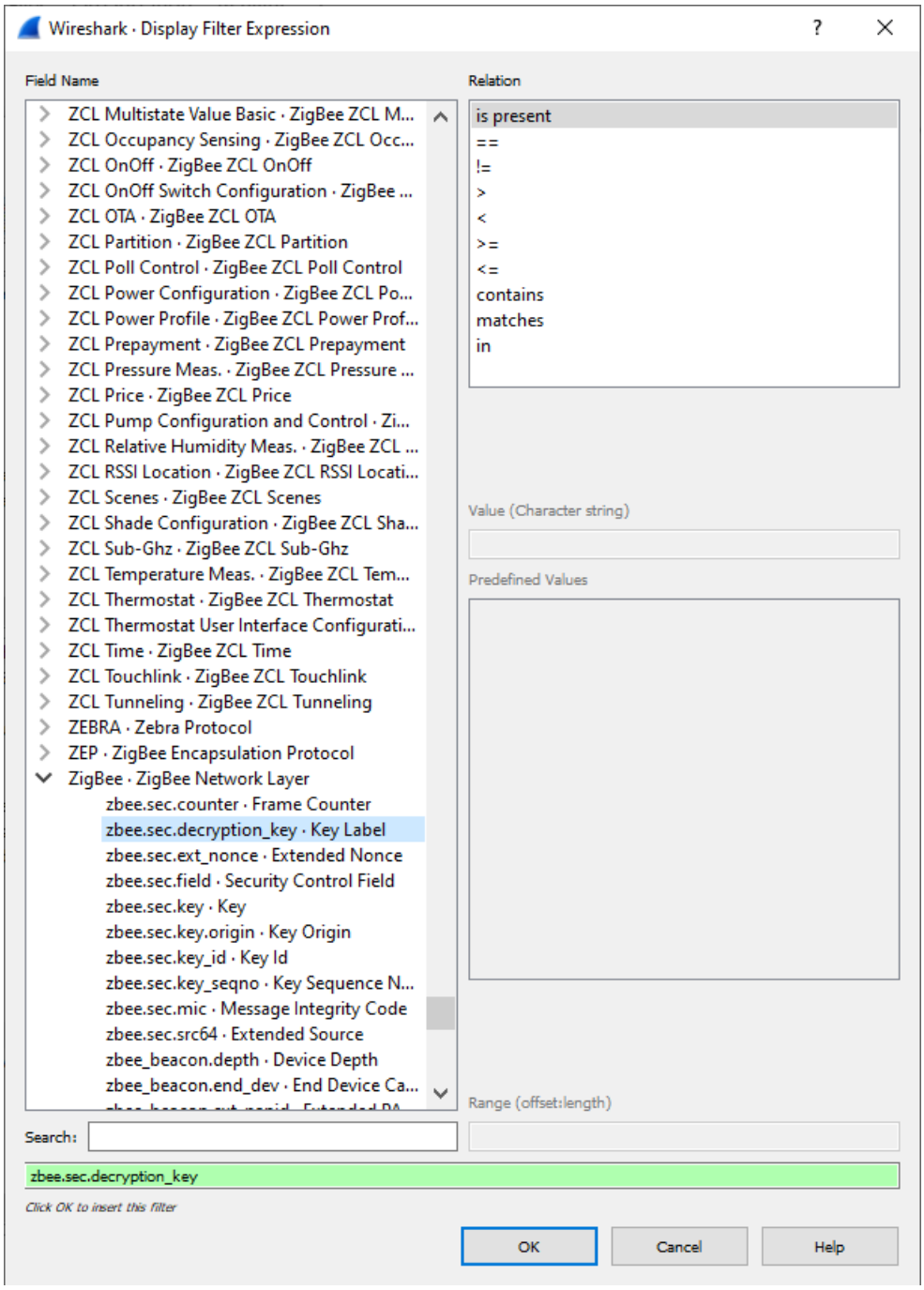



*Figure 5: Wireshark interface for expressions*

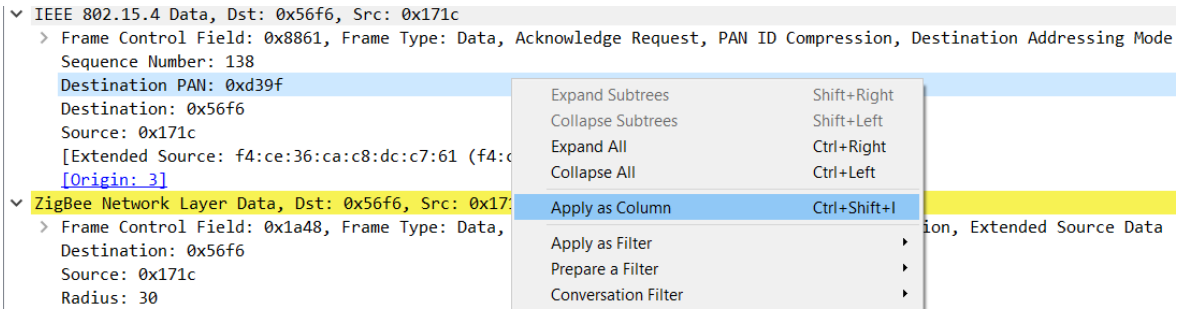
See the following table for some filter examples.

| Display filter | Description |
|---|---|
| wpan | Filter that displays all IEEE 802.15.4 traffic. |
| wpan.dst_pan | Filter that displays IEEE 802.15.4 packets that have a specific destination PAN. |
| wpan.dst16 | Filter that displays short destination addresses of IEEE 802.15.4 frames. |
| wpan.dst64 | Filter that displays long destination addresses of IEEE 802.15.4 frames. |
| wpan.src16 | Filter that displays short source addresses of IEEE 802.15.4 frames. |
| wpan.src64 | Filter that displays long source addresses of IEEE 802.15.4 frames. |
| ipv6, coap, dtls, udp | Examples of filters for packets that can be encountered on Thread and IP networks. |
| mle | Protocol filter that displays all Mesh Link Establishment traffic. Used for example by Thread. |

*Table 2: Display filtering*

The following tips can help when inspecting your data:

- Turn any field in the **packet details pane** into a column. To do so:
  a) Right-click the value in the packet details.
  b) Click **Apply as Column**.




- Apply a value as a filter to, for example, see only operations affecting a particular handle. To filter packets that have a specific value for some field:
  a) Right-click the value in the packet details.
  b) Click **Apply as Filter**.
  c) Click **Selected**.
- Save a set of captured packets to be able to look at them later. To do so:
  a) Click the **Stop** button to stop capturing packets.
  b) Click **File** > **Save As** to save all packets, or click **File** > **Export Specified Packets** to save a selection of packets.
- Clear the packet list and restart a capture by clicking the **Restart** button.

See the documentation on the Wireshark website for more information.

# 6 Troubleshooting

If you have problems installing or using the nRF Sniffer for 802.15.4, see the following sections for troubleshooting information.

## nRF Sniffer for 802.15.4 is not listed in the Wireshark interface

Check that the hardware is set up correctly:

1. Ensure that the development kit or dongle has been recognized as a USB device and that the drivers are loaded.
2. Ensure that the firmware HEX file has been programmed.
3. Reset the hardware by unplugging the hardware, waiting 5 seconds, and plugging it back in.

If these steps do not help, verify that you have installed the nRF Sniffer capture tool correctly, including the pySerial module, and that the files located in the extcap folder can be run as described in the capture tool verification step.

## nRF Sniffer for 802.15.4 is not listed in the Wireshark interface on Linux despite correct capture tool installation

Make sure you have sufficient permissions to access the serial device and that Wireshark is available to all users, not only the root user, as described in Installing Wireshark on Ubuntu Linux on page 6.

## nRF Sniffer for 802.15.4 is not listed in the Wireshark interface on Windows despite correct capture tool installation

Make sure that the Python installation directory is added to the environment variables.

After you add the directory to the environment variables, verify that the configuration is correct:

1. Open a command window.
2. Run the `python` command followed by `import serial`. For example:

```
# python
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 09:44:33) [MSC v.1900 32 bit (Intel)]
 on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
```

If there are no errors, the configuration is correct.

## nRF Sniffer for 802.15.4 capture stops when another Wireshark process is started on Linux

Because Linux applications primarily use advisory locking, there is nothing stopping other applications from opening and writing data to a serial port if you have other extcap plugins installed. For instance, the nRF Sniffer for Bluetooth LE extcap plugin discovers connected *Bluetooth*® Low Energy sniffers at Wireshark startup by actively sending data to all serial ports. This can cause unexpected behavior of the nRF Sniffer for 802.15.4.

To avoid this issue, make sure that the other extcap plugins do not send any data to serial ports.

## nRF Sniffer for 802.15.4 does not respond when starting capturing in Wireshark on Windows

This is an issue affecting the latest releases of Wireshark. Wait for around 15 seconds for the capture to start. Alternatively, you can try installing an older version of Wireshark:

1. Go to Wireshark download page.
2. In the **Old Stable Release** section, click the release package name for your operating system version.
3. Download and install the tool.

You can also download one of the older release versions from the Go Spelunking section that contains links to mirrors that store all previous versions of Wireshark.

## nRF Sniffer for 802.15.4 does not receive packets in Wireshark

Make sure you specified the correct channel for capturing packets. To do so, click the gear icon next to the hardware interface name when running nRF Sniffer and specify the channel in the **Interface Options** window. The channel depends on the application configuration.

## nRF Sniffer for 802.15.4 capture stops with ModemManager service enabled on Linux

On some occasions, the ModemManager service may send AT commands to the nRF Sniffer for 802.15.4.

To prevent this situation from happening, use one of the following options:

- Disable the ModemManager service. To do so, complete the following steps:
  1. Stop the service by typing: `sudo systemctl stop ModemManager.service`
  2. Disable the service by typing: `sudo systemctl disable ModemManager.service`
- If the ModemManager service runs either on **`DEFAULT`** or **`PARANOID`** policy, create an **`udev`** rule. To do so:
  1. Create a new file `99-mm-blacklist.rules` in the `/etc/udev/rules.d/` folder, with the following configuration:

     ```
     ACTION!="add", SUBSYSTEM!="usb_device", GOTO="mm_blacklist_rules_end"

     ATTR{idProduct}=="154a", ATTR{idVendor}=="1915", ENV{ID_MM_DEVICE_IGNORE}="1"

     LABEL="mm_blacklist_rules_end"
     ```
  2. Apply the new **`udev`** rules by typing: `udevadm trigger`.
  3. Verify that the settings have been successfully applied by typing: `udevadm info -q property -n /dev/ttyACMx`. The following return values confirm that the settings are correctly applied:

     ```
     ID_MM_CANDIDATE=0
     ID_MM_DEVICE_IGNORE=1
     ```
  4. Restart the ModemManager service by typing: `sudo systemctl restart ModemManager`.

## nRF Sniffer for 802.15.4 does not receive packets on Windows

On Windows, COM port numbers higher than 199 are not supported. If the COM port number is COM200 or higher, rename the COM port on Windows to a COM port number that is COM199 or lower. To do so, complete the following steps:

1. Open the Device Manager and click **Ports (COM & LPT)**.
2. Right-click on your COM port and click **Properties**.

3. In Properties, go to the **Port Settings** tab and click **Advanced**.
4. Change the COM port number by clicking the COM port number drop-down and selecting a COM port that is less than 200. Select a COM port number that is not in the list of devices currently attached to your computer. These are listed in the Device Manager under **Ports (COM & LPT)**.
5. Click **OK** and accept the changes when asked whether you want to continue.

# Legal notices

By using this documentation you agree to our terms and conditions of use. Nordic Semiconductor may change these terms and conditions at any time without notice.

## Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function, or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Nordic Semiconductor ASA does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. If there are any discrepancies, ambiguities or conflicts in Nordic Semiconductor's documentation, the Product Specification prevails.

Nordic Semiconductor ASA reserves the right to make corrections, enhancements, and other changes to this document without notice.

## Life support applications

Nordic Semiconductor products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury.

Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

## RoHS and REACH statement

Complete hazardous substance reports, material composition reports and latest version of Nordic's REACH statement can be found on our website www.nordicsemi.com.

## Trademarks

All trademarks, service marks, trade names, product names, and logos appearing in this documentation are the property of their respective owners.

## Copyright notice