nRF Util

v7.1.0

User Guide

4399_018 / 2023-02-22

NORDIC SEMICONDUCTOR

Contents

Revision history

Date	Description
2023-02-22	Updated: • Introduction on page 4 introduction is updated with new commands **`trace`** and **`completion`** • New chapter added for Tracing on page 14 • Programming devices on page 10 renamed and extended with device-specific programming use cases • Editorial changes
2022-11-07	First release

Previous versions

PDF files for relevant previous versions are available here:

- nRF Util User Guide v7.0.0

1 Introduction

nRF Util is a modular command line tool, enabling power users to manage Nordic Semiconductor devices and support automation.

nRF Util is provided as an executable file (`nrfutil`) that can install, manage, and launch additional domain specific commands. nRF Util supports JSON output, for automation, scripting, and custom manipulation of the data available from nRF Util. The underlying architecture of nRF Util is aligned with that of nRF Connect for Desktop applications.

This user guide helps you get started, with information common to all the commands and device-specific use cases for the **`device`** and **`trace`** commands. For additional information, use the built-in help system and see the DevZone nRF Util blog.

nRF Util supports the following commands:

- **`device`**
  - For device discovery, programming, and operations such as erase, reset, and recover, use the **`nrfutil device`** command.
- **`trace`**
  - To analyze or monitor an application's communication or power consumption, use the **`nrfutil trace`** command to collect trace data from Nordic Semiconductor devices.
- **`completion`**
  - To support auto-completion for the bash, powershell, and zsh shells install the **`nrfutil completion`** command. See nRF Util setup auto-completion.
- **`nrf5sdk-tools`**
  - This command supports *Device Firmware Update (DFU)* features of the Python based nRF Util (version 6.1.0 and lower), designed for nRF5 SDK.
  - To install and use this command, please refer to nRF Util for nRF5 SDK.

nRF Util has tier 1 support for the following platforms:

- Ubuntu 20.04 LTS (x64)
- Windows 10 (x64)
- MacOS 11 (x64, arm64)

In addition, nRF Util has tier 2 support for Windows 11 (x64), Ubuntu 22.04 (x64), and MacOS 12 (x64, arm64).

You can find the tier definitions here: nRF Connect SDK Supported Operating Systems.

2 Minimum requirements

Before you start using nRF Util, check that you have the required software.

- SEGGER J-link - required for all platforms. Download the installer for your platform from SEGGER J-Link Software.
- Microsoft Visual C++ Redistributable - required for Windows. Follow the instructions at Microsoft Visual C++ Redistributable.
- libusb-1.0 and nRF-udev - required for Linux. See Installing libusb-1.0 and nRF-udev on Linux on page 5.

2.1 Installing libusb-1.0 and nRF-udev on Linux

libusb-1.0 usually comes installed with Ubuntu and nrf-udev can be installed by downloading a .deb file from the nrf-udev project.

1. Run the command `sudo apt install libusb-1.0-0`
2. Download the latest `.deb` from nRF-udev.
3. Run: `sudo dpkg -i nrf-udev_1.0.1-all.deb`

3 Installing nRF Util

nRF Util functionality is provided through modular, domain specific commands that are installed from a central package registry on the Internet.

The nrfutil executable file is a helper application which helps you search for, install, list, and upgrade the modular commands. Download and use it to find and install the command tools you need.

1. Download the nrfutil executable file from nRF Util.
2. Install nRF Util by doing one of the following:
   - Run nrfutil from the command line in the folder where it was downloaded.
   - Move it to a folder in the system path, to run it from anywhere on the system.
3. On macOS and Linux, give nrfutil execute permissions by typing `chmod +x nrfutil` in a terminal or using a file browser.

   This is typically a checkbox found under file properties.
4. Now you are ready to install commands. To list available commands run **`nrfutil search`**
5. To install a command, run **`nrfutil install <command-name>`**.

The command package is installed to a sub folder under `{USER HOME DIRECTORY}/.nrfutil/`.

For example, to install the **`device`** command, run the following:

```
nrfutil install device
```

You can now use the command **`nrfutil device`**, for example with --help.

4 Getting Started

Common nRF Util procedures such as help, managing output, upgrading, and offline installation are described here.

Some useful commands and their descriptions are summarized in the following table:

Run	Description
`nrfutil --help`	To view `nrfutil` subcommands and get help on these.
`nrfutil search`	To view a list of installable commands, available on the central package registry on the Internet.
`nrfutil list`	To view a list of all commands that are installed on your host.
`nrfutil device --help`	To list and get help on a command's sub-commands **`device`** is used in this example.
`nrfutil install --help`	To get help on the `nrfutil install` sub-command.
`nrfutil --version`	To view the version of `nrfutil`.
`nrfutil device --version`	To view the version of the `device` command.

Table 1: nRF Util commands and subcommands

4.1 Displaying help

Add `--help` to any nRF Util command to display help about the command.

The following command line context-sensitive help is available for commands and subcommands:

- To view complete help, append `--help` to the command or subcommand.
- To view a concise version, append `-h` instead.

See some examples in the following table:

Run	Description
nrfutil --help	To view help on the nRFutil command.
`nrfutil device -help`	To view concise help on the device command.
`nrfutil device list --help`	To view help on the nrfutil device list subcommand.

Table 2: Useful help commands

If you need usage instructions while writing a command, you can use `-h` and `--help`, regardless of any other arguments supplied. For example,

```
nrfutil device program --firmware firmware.hex --options --help
```

shows the usage instructions for **`nrfutil device program`**.

4.2 Managing command output with JSON

Use *JavaScript Object Notation (JSON)* output for scripting and automation when building applications with nRF Util functionality.

nRF Util supports JSON output through the command line flag `--json`. This structures the data in a way that can be parsed by JSON parsers, available in many programming and scripting languages.

Data from nRF Util is communicated as a sequence of JSON messages. Each message is encoded as a single JSON object on a single line. All messages follow a common format:

```
{"type":[task_begin|task_end|task_progress|info|log]
"data": { ... }}
```

To view Python examples of using the JSON output from `nrfutil device`, run `nrfutil device --help-extended`.

Message types:

- `task_begin`: Indicates that a task with progress status updates has started.
- `task_end`: Indicates that a task has ended.
- `task_progress`: Indicates a progress update to a task.
- `info`: Indicates a general info message containing arbitrary data.
- `log`: Indicates a log message, emitted when `--log-output=stdout` is supplied.

The `task_*` messages correspond to the progress bars and progress spinners rendered when not using the `--json` flag and are used to indicate that a long-running task is in progress.

The `info` message is used to convey general information. For example, `nrfutil search --json` uses the `info` message to convey the list of installable commands in JSON, and `nrfutil device list --json` uses the `info` message to list device information in JSON. To output the data payload only, add the flag `--skip-overhead`.

The main data payload in the JSON messages is contained within the `data` field. The task messages share some common fields across different contexts. The fields for the `info` messages vary from context to context. Investigate the fields manually when building applications on top of the JSON messages.

Standard semantic versioning guarantees are given for the JSON output formats across the different commands. Releases of the commands within the same major version number have backwards-compatible JSON messages.

4.3 Upgrading nRF Util commands

You can view commands with available updates by running `nrfutil list --outdated`.

When updates are available, you can update commands by running:

```
nrfutil upgrade
```

4.4 Installing nRF Util when offline

For offline distribution of nRF Util and its installable commands, use the `prepare-offline` subcommand.

The following steps show how to install offline from a USB drive. It assumes Windows maps the USB drive to `E:/`.

1. When connected to the Internet, copy the nRF Util executable to the USB drive and run the following:

   ```
   nrfutil prepare-offline E:/nrfutil-offline
   ```

   This downloads and prepares tarballs containing the latest version of all available nRF Util commands to the USB drive, along with a portable package index that nRF Util uses for querying.
2. To install, for example, the **`device`** command on a host without an Internet connection, connect the USB drive and run the following:

   ```
   nrfutil install device --from-offline E:/nrfutil-offline
   ```

When new versions are available, you can upgrade installed commands by running the following:

```
nrfutil upgrade --from-offline E:/nrfutil-offline
```

5 Programming and managing devices with nrfutil device

For device discovery, programming, and operations such as erase, reset, and recover, use the **`nrfutil device`** command.

nRF Util supports Nordic Semiconductor *Development Kit (DK)*s, prototyping platforms, and dongles.

To view a list of **`device`** operations, run `nrfutil device --help`. See also Displaying help on page 7. The following sections cover common use cases.

5.1 Listing device serial numbers and traits

When programming and tracing, you must specify the target device or devices. This can be done using the device serial number or traits, which are discovered using **`nrfutil device list`**.

All Nordic Semiconductor devices are annotated with traits, which are used to determine whether a device supports a given device operation or programming method.

Run	Description
`nrfutil device list`	To view a list of connected devices and information such as serial numbers, com ports and traits.
`nrfutil device list --json-pretty`	To view more information on the devices.
`nrfutil device list --help`	To view a full list of available traits and their descriptions.
`nrfutil device list --traits jlink`	To filter devices based on a trait, in this example `jlink`

Table 3: `nrfutil device list` commands

The following example shows how to program new firmware to a device using its serial number:

```
nrfutil device program --serial-number xxxxxxxxx --firmware /path/fw.hex
```

To program firmware to multiple devices in a batch operation, you can supply multiple serial numbers:

```
nrfutil device program --serial-number xxxxxxxxx,yyyyyyyyy --firmware /path/fw.hex
```

Alternatively, devices can be selected based on their traits:

```
nrfutil device program --traits jlink --firmware /path/fw.hex
```

5.2 Programming devices

`nrfutil device` uses the same architecture and programming methods as the nRF Connect for Desktop Programmer app for listing, programming, and other device operations.

All programming operations are exposed through **`nrfutil device program`**. The command is the same regardless of whether the programming is over SEGGER J-Link, MCUBoot serial recovery, nRF9160 modem upgrade, or other. nRF Util automatically selects programming method based on the device traits and the firmware file format.

5.2.1 Programming a device with a SEGGER J-Link OB debugger

Nordic Semiconductor DKs contain SEGGER J-Link *Onboard (OB)* debuggers and are listed with the `jlink` device trait when connected to the computer through the J-Link USB port.

You can use the following procedure to progam new firmware to the board:

1. Find the serial number of the J-Link device connected to the system, by running:

   ```
   nrfutil device list --traits jlink
   ```

2. Delete old firmware on the board, by running the following:

   ```
   nrfutil device erase --serial-number xxxxxxxxx
   ```

3. Program the new firmware to the board, by running:

   ```
   nrfutil device program --serial-number xxxxxxxxx --firmware /path/fw.hex
   ```

4. Reset the device, by running:

   ```
   nrfutil device reset --serial-number xxxxxxxxx
   ```

Alternatively, you can use the `--options` flag to combine the previous steps, as follows:

```
nrfutil device program --serial-number xxxxxxxxx --firmware /path/fw.hex --options
 chip_erase_mode=ERASE_ALL,reset=RESET_SYSTEM
```

5.2.2 Programming an nRF52840 Dongle over Nordic Secure DFU

Nordic Secure *DFU* is made over a UART serial connection.

A DFU package is required, see Generating DFU packages in the nRF Util for nRF5 SDK User Guide.

Nordic Secure DFU is the primary programming method for the nRF52840 dongle. The following procedure replaces the DFU over USB serial connection in the nRF Util for nRF5 SDK command.

1. List the devices with a DFU trigger interface, by running:

   ```
   nrfutil device list --traits nordicDfu
   ```

   If the device does not show up with the expected trait, it might need to be put into programming mode by holding in the reset button and reconnecting the dongle to the system.

2. Program new firmware to a device identified by its serial number, by running:

   ```
   nrfutil device program --firmware dfu_package.zip --serial-number xxxxxxxxxxxx
   ```

   The device is automatically put into application mode after programming. This behavior can be controlled using the `--options` flag, see:

   ```
   nrfutil device program --help
   ```

5.2.3 Upgrading the modem firmware of an nRF9160 DK

`nrfutil device` can be used to upgrade modem firmware on nRF9160 DKs. These devices are listed using the `jlink` and `modem` traits.

Modem firmware files are available at nRF9160 DK Downloads.

1. Identify the serial number of J-Link devices that support modem upgrades, by running the following:

   ```
   nrfutil device list --traits modem,jlink
   ```

   The nRF9160 DK should show up as a device with board version PCA10090.
2. To program, for example, the firmware `mfw_nrf9160_1.3.3.zip`, run the following:

   ```
   nrfutil device program --firmware mfw_nrf9160_1.3.3.zip --serial-number xxxxxxxxx
   ```

5.2.4 Programming application firmware on a Nordic Thingy:91

These devices are listed using the `mcuboot` trait.

Build your nRF Connect SDK project for the nRF9160 *System in Package (SiP)* of the Nordic Thingy:91™, for example, the thingy91_nrf9160_ns and use the file `build/zephyr/app_signed.hex` generated, see Multi-image builds

1. Find the device serial number by running the following:

   ```
   nrfutil device list --traits mcuboot
   ```
2. Before programming, put the nRF9160 SiP of the Nordic Thingy:91 in application serial recovery mode by holding in the **SW3** button while powering it off and on.
3. Program `build/zephyr/app_signed.hex` to a device with serial number `THINGY91_xxxxxxxxxxxx`, by running the following:

   ```
   nrfutil device program --firmware build/zephyr/app_signed.hex --serial-number
    THINGY91_xxxxxxxxxxxx
   ```

If **`nrfutil device`** fails with a timeout error during step 3, put it into application serial recovery mode as described in step 2. See also Troubleshooting on page 19.

5.2.5 Upgrading the modem firmware of a Nordic Thingy:91 over MCUBoot serial recovery

The traits `mcuBoot` and `modem` indicate that the Nordic Thingy:91 supports modem upgrades over MCUBoot serial recovery.

Download the nRF9160 *SiP* modem firmware from Nordic Thingy:91 Downloads.

1. List the devices connected to the system by running the following:

   ```
   nrfutil device list --traits mcuboot,modem
   ```
2. Put the Nordic Thingy:91 in application serial recovery mode by holding in the **SW3** button while powering it off and on.
3. Upgrade the modem firmware by running:

   ```
   nrfutil device program --firmware mfw_nrf9160_1.3.3.zip --serial-number
    THINGY91_xxxxxxxxxxxx
   ```

5.2.6 Programming the application firmware on a Nordic Thingy:53

Nordic Thingy:53™ devices in application serial recovery mode are annotated with the `mcuBoot` trait.

Precompiled application firmware samples are available at Nordic Thingy:53 Downloads. These are multi-image MCUBoot DFU zip packages.

1. To find the device serial number, list devices with the `mcuBoot` trait:

```
nrfutil device list --traits mcuboot
```

2. To program, for example, the machine learning sample on a device with serial number `xxxxxxxxxxxxxxxx`, run:

```
nrfutil device program --firmware Machine_Learning/
machine_learning_2.0.0_thingy53_nrf5340_zrelease.zip --serial-number xxxxxxxxxxxxxxx
```

In nRF Connect SDK, samples compatible with the Nordic Thingy:53, such as the machine learning sample, building for the `thingy53_nrf5340_cpuapp_ns` target generates a file `build/zephyr/dfu_application.zip`. This corresponds to the zip package previously used and can be programmed in the same way:

```
nrfutil device program --firmware build/zephyr/dfu_application.zip --serial-number
 xxxxxxxxxxxxxxx
```

The file `build/zephyr/app_signed.hex` can also be used:

```
nrfutil device program --firmware build/zephyr/app_signed.hex --serial-number
 xxxxxxxxxxxxxxx
```

6 Tracing

To analyze or monitor an application's communication or power consumption, use the **`nrfutil trace`** command to collect trace data from Nordic Semiconductor devices.

The tool collects *Universal Asynchronous Receiver/Transmitter (UART)* traces over the serial port. It supports the following protocols:

- *AT command*
- *Internet Protocol (IP)*
- *LTE Radio Resource Control (LTE-RRC)*
- *Non-Access-Stratum protocol for Evolved Packet System (NAS-EPS)* IOT broadcast

nRF Util traces can be saved to file or streamed from a serial port for analysis and monitoring. Multiple trace formats are supported, and traces can be exported to software, such as *Wireshark* or the Nordic Semiconductor *Online Power Profiler (OPP)*. The saved trace files can be transformed from raw binary to alternative formats.

To view a list of **`trace`** operations, run `nrfutil trace --help`. See also Displaying help on page 7.

The following sections cover some device-specific use cases and **`trace`** configuration information.

6.1 Minimum requirements

Before you start using **`nrfutil trace`**, ensure the **`device`** command is installed.

You need device information, such as serial ports, serial numbers, and traits, for tracing. See Installing nRF Util on page 6 and Listing device serial numbers and traits on page 10.

6.2 Capturing trace data from an nRF9160 modem to a binary file

nRF Util **`trace`** can be used to collect *UART* traces from the nRF9160 *SiP* or any device supporting modem trace. The trace can be used to analyze the communication between the device and the *Long-Term Evolution (LTE)* network.

1. Identify the device's serial port for tracing:

   ```
   nrfutil device list --traits modem
   ```

   When selecting a serial port for the trace, pick the serial port with the highest vcom value.

2. To trace continuously, for example, on `COM10` and output to *Packet Capture Next Generation (PcapNG)* format, run the following:

   ```
   nrfutil trace lte --input-serialport COM10 --output-pcapng live-trace.pcapng
   ```

3. To stop tracing, press **`CTRL+C`**.

   The trace file is created in the current directory unless a different path is specified. You can view the file size in bytes while trace is running.

Alternatively, the trace can be streamed directly to *Wireshark* by running:

```
nrfutil trace lte --input-serialport COM10 --output-wireshark path_to_wireshark
```

6.3 Collecting a modem trace for the nRF9160 DK

To collect a modem trace, you must ensure that you have the latest firmware for the board controller, update your application to enable tracing, and capture the trace while your application is running.

6.3.1 Preparing the board for trace collection

The nRF Util requires a current version of the board controller firmware to be programmed on the nRF52840 *System on Chip (SoC)* of the nRF9160 DK.

Download the latest firmware from nRF9160 DK Downloads (scroll down to **Board controller firmware**).

Complete the following steps to program the board controller firmware:

1. Set the switch that configures which chip to program to the **nRF52** position.

   This switch is labeled **PROG/DEBUG** (**SW10** on nRF9160 DK v0.15.0 and later, **SW5** on earlier versions).
2. Connect your device to the computer with a *Universal Serial Bus (USB)* cable and power it on or reset it if it is already connected.
3. Program the downloaded firmware to your device.
   - Using nRF Util `device`, program the downloaded firmware to your device.

     ```
     nrfutil device program -serial-number <SERIAL_NUMBER> --firmware <FIRMWARE>
     ```

6.3.2 Enabling tracing in the application

To capture a modem trace, you must configure your application to enable trace output over *UART*.

The following instructions assume that your application is based on the nRF Connect SDK versions later than 2.0.0.

> **Note:** By default, nRF Connect SDK's modem library uses the UART1 peripheral for trace output. This means that you cannot use UART1 for other purposes in your application. If this does not work for your application, you must update the configuration and code of the modem library to use a different UART peripheral for trace output.

Complete the following steps to enable tracing:

1. Set the CONFIG_NRF_MODEM_LIB_TRACE option in your application.

   See Configuring your application for instructions on how to set this option temporarily or permanently.

   > **Note:**
   > - In nRF Connect SDK versions 1.5.0 - 2.0.0, the option was called CONFIG_NRF_MODEM_LIB_TRACE_ENABLED.
   > - In nRF Connect SDK v1.5.x, setting the option temporarily might cause a build error. In that case, set the option permanently in the `prj.conf` file.
   > - In nRF Connect SDK versions before 1.5.0, the option was called CONFIG_BSD_LIBRARY_TRACE_ENABLED.
2. Set the switch that configures which chip to program to the **nRF91** position.

   This switch is labeled **PROG/DEBUG** (**SW10** on nRF9160 DK v0.15.0 and later, **SW5** on earlier versions).

3. Connect your device to the computer with a *USB* cable and power it on or reset it if it is already connected.
4. Build your application and program it to the device.

   Follow the instructions in Building and programming a sample application.

6.3.3 Capturing the modem trace

The modem trace is captured using the **`nrfutil trace lte`** subcommand.

1. Identify the device's serial port for tracing:

   ```
   nrfutil device list --traits modem
   ```

   When selecting a serial port for the trace, pick the serial port with the highest vcom value.
2. To trace continuously, for example, on `COM10` and output to *PcapNG* format, run the following:

   ```
   nrfutil trace lte --input-serialport COM10 --output-pcapng live-trace.pcapng
   ```

3. To stop tracing, press **`CTRL+C`**.

   The trace file is created in the current directory unless a different path is specified. You can view the file size in bytes while trace is running.

Alternatively, the trace can be streamed directly to *Wireshark* by running:

```
nrfutil trace lte --input-serialport COM10 --output-wireshark path_to_wireshark
```

6.4 Collecting a modem trace for a Nordic Thingy:91

To collect a modem trace, you must ensure that you have the latest firmware for the board controller, update your application to enable tracing, and capture the trace while your application is running.

6.4.1 Preparing the board for trace collection

The nRF Util requires a current version of the board controller firmware to be programmed on the nRF52840 *SoC* of the Nordic Thingy:91.

Download the latest firmware from Nordic Thingy:91 Downloads (select **Precompiled application and modem firmware**).

The archive contains images for different applications in different formats. Choose an image based on the method you use to update the firmware:

- When programming through an external debug probe, follow the steps in Updating firmware through external debug probe to program the nRF52840 *SoC*.
- When programming through *USB*, follow the steps in Updating firmware through USB to program the nRF52840 *SoC*.

6.4.2 Enabling tracing in the application

To capture a modem trace, you must configure your application to enable trace output over *UART*.

The following instructions assume that your application is based on the nRF Connect SDK versions later than 2.0.0.

Note: By default, nRF Connect SDK's modem library uses the UART1 peripheral for trace output. This means that you cannot use UART1 for other purposes in your application. If this does not work for your application, you must update the configuration and code of the modem library to use a different UART peripheral for trace output.

Complete the following steps to enable tracing:

1. Set the CONFIG_NRF_MODEM_LIB_TRACE option in your application.

   See Configuring your application for instructions on how to set this option temporarily or permanently.

   Note:

   - In nRF Connect SDK versions 1.5.0 - 2.0.0, the option was called CONFIG_NRF_MODEM_LIB_TRACE_ENABLED.
   - In nRF Connect SDK v1.5.x, setting the option temporarily might cause a build error. In that case, set the option permanently in the `prj.conf` file.
   - In nRF Connect SDK versions before 1.5.0, the option was called CONFIG_BSD_LIBRARY_TRACE_ENABLED.

2. Build your application and program it to the device.

   - When programming through an external debug probe, set the switch that configures which chip to program (**SW2**) to the **nRF91** position and follow the instructions in Building and programming from the source code.
   - When programming through *USB*, put the device into application serial recovery mode by pressing and holding the **SW3** button while powering on. Follow the instructions in Getting started with Thingy:91 to program the application.

6.4.3 Capturing the modem trace

The modem trace is captured using the **`nrfutil trace lte`** subcommand.

1. Identify the device's serial port for tracing:

   ```
   nrfutil device list --traits modem
   ```

   When selecting a serial port for the trace, pick the serial port with the highest vcom value.

2. To trace continuously, for example, on `COM10` and output to *PcapNG* format, run the following:

   ```
   nrfutil trace lte --input-serialport COM10 --output-pcapng live-trace.pcapng
   ```

3. To stop tracing, press **`CTRL+C`**.

   The trace file is created in the current directory unless a different path is specified. You can view the file size in bytes while trace is running.

Alternatively, the trace can be streamed directly to *Wireshark* by running:

```
nrfutil trace lte --input-serialport COM10 --output-wireshark path_to_wireshark
```

6.5 Choosing and transforming trace output format

The output format can be selected when starting a trace and a binary trace file can be transformed afterwards.

Option	Description
`--output-raw  filename.bin`	To export as a binary trace file. Raw files are primarily used as an attachment when assistance is needed from Nordic Semiconductor support.
`--output-wireshark path_to_wireshark`	To pipe the output to *Wireshark*, specify the path to a Wireshark executable.
`--output-pcapng out.pcapng`	To convert the output to a file with pcapng format.
`--output-opp config.json`	For use with *OPP*.

Table 4: Trace output format options

The following example shows how to transform a binary file `raw-file.bin` to *PcapNG*:

```
nrfutil trace lte --input-file raw-file.bin --output-pcapng out.pcapng
```

6.6 Viewing nRF91 LTE modem power consumption

To view the power consumption of the nRF91 LTE modem, you can convert trace data to JSON format which can be read by the Nordic Semiconductor *OPP*.

Use the nRF Util **`trace`** command with the `--output-opp` option, as follows:

```
nrfutil trace lte <input-options> --output-opp config.json
```

The configuration file generated can be imported to the OPP. For more information see Online Power Profiler LTE user guide.

6.7 Customizing the trace configuration

To change to the serial port settings, such as baud rate, stop, or parity bits, you can modify the trace configuration.

The **`nrfutil trace lte`** subcommand creates a temporary trace configuration file which is discarded when tracing has finished. You can export a copy of the configuration file and modify it as follows.

1. When tracing, pass the flag `-keep-configuration`.
   When tracing has completed, a file with the naming style `trace-collection-configuration-YYYYMMDD-ThhmmssUTC.json` is produced.
2. Edit the file and change the required settings.
3. Pass the updated configuration to the **`nrfutil trace process`** subcommand as follows:

   ```
   nrfutil trace process --config trace-collection-configuration-YYYYMMDD-ThhmmssUTC.json
   ```

7 Troubleshooting

Refer to this section for nRF Util **`device`** and **`trace`** troubleshooting tips.

nRF52840 dongle not found

The dongle is listed with the trait `nordicDfu`. If not found, put it into programming mode by holding in the reset button and reconnecting the dongle to the system.

Nordic Thingy:91 not found

When programming the application firmware for the nRF9160 *SiP* of a Nordic Thingy:91, make sure that you build the application for the nRF9160 SiP, and put the Nordic Thingy:91 in application serial recovery mode by holding in the **SW3** button while powering it off and on. If **`nrftil device`** fails with a timeout error during the erase step, put it into application serial recovery mode using the previous procedure.

> **Note:** The device can become unusable if, while programming, you use the wrong button and put it into the application serial recovery mode for the nRF52840 *SoC*. In this case, it no longer appears in `nrfutil device list`, and you must reprogram the nRF52 connectivity bridge firmware from the Nordic Thingy:91 download page through the external debug probe of the nRF9160 *DK*. See the programming instructions in Thingy:91 firmware images.

Nordic Thingy:53 not found

If the Nordic Thingy:53 device does not show up, put it in application serial recovery mode by holding in the **SW2** button while powering the device off and on. You can verify that the Nordic Thingy:53 is in bootloader mode by running `nrfutil device list --json` and verifying that the product name of the device is `Bootloader Thingy:53`.

No output when tracing

- Make sure that your modem firmware is up to date:
  - nRF9160 DK Downloads
  - Thingy:91 firmware images
- Check that your device's application has modem tracing enabled.
- Note that the KConfig flag for modem tracing varies depending on nRF Connect SDK version.
- Check the serial port used for **`trace`**. Normally the serial port with the highest VCOM value should be chosen. Make sure that your application does not allocate this serial port for anything else.

Glossary

AT command

A command used to control the modem.

Device Firmware Update (DFU)

A mechanism for upgrading the firmware of a device.

Development Kit (DK)

A hardware development platform used for application development.

Integrated Development Environment (IDE)

A software application that provides facilities for software development.

Internet Protocol (IP)

The network layer communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

JavaScript Object Notation (JSON)

A lightweight data-interchange format.

Long-Term Evolution (LTE)

A wireless broadband communication standard for mobile devices and data terminals, based on the GSM/EDGE and UMTS/HSPA technologies.

LTE Radio Resource Control (LTE-RRC)

A protocol that controls MAC, RLC, and PDCP layers in the LTE air interface, connecting an LTE UE with an eNodeB.

Non-Access-Stratum protocol for Evolved Packet System (NAS-EPS)

A protocol used for LTE mobility and session management.

Onboard (OB)

A function that is delivered on the chip microcontroller.

Online Power Profiler (OPP)

A tool for estimating LTE-M, NB-IoT, or *Bluetooth*® Low Energy current consumption.

Packet Capture Next Generation (PcapNG)

The default file format for Wireshark.

System in Package (SiP)

Several integrated circuits, often from different technologies, enclosed in a single module that performs as a system or subsystem.

SEGGER Embedded Studio (SES)

A cross-platform *Integrated Development Environment (IDE)* for embedded C/C++ programming with support for Nordic Semiconductor devices, produced by SEGGER Microcontroller.

System on Chip (SoC)

A microchip that integrates all the necessary electronic circuits and components of a computer or other electronic systems on a single integrated circuit.

Universal Asynchronous Receiver/Transmitter (UART)

A hardware device for asynchronous serial communication between devices.

Universal Serial Bus (USB)

An industry standard that establishes specifications for cables and connectors and protocols for connection, communication, and power supply between computers, peripheral devices, and other computers.

Wireshark

A free software tool that captures wireless traffic and reproduces it in a readable format. It is a cross-platform network protocol analyzer that can be used to view, analyze, and troubleshoot packets sent over a data network.

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