nrfutil

v3.4.x

User Guide

v1.3

NORDIC® SEMICONDUCTOR

Contents

Revision history

Date	Version	Description
December 2017	1.3	Updated for nrfutil v3.4.0: • Added Performing a DFU over Thread on page 9 • Added Performing a DFU over a serial USB connection on page 11 • Updated the examples for Generating DFU packages on page 8 to show how to create unsigned packages
March 2017	1.2	Updated for nrfutil v2.2.0: • Added support for serial DFU (Performing a DFU over a serial UART connection on page 10) • Changed the location of the `dfu-cc.proto` file in the DFU bootloader implementation (Customizing the init packet on page 15)
November 2016	1.1	Updated for nrfutil v2.0.0
September 2016	1.0	First release

1 nrfutil

nrfutil is a Python package and command-line utility that supports Device Firmware Updates (DFU) and cryptographic functionality.

See the DFU bootloader and BLE Secure DFU Bootloader example in the SDK documentation for more information about Device Firmware Updates. The nrfutil application and its library offer the following features:

- DFU package generation
- Cryptographic key generation, management, and storage
- Bootloader settings generation
- DFU procedure over any of the following transports:
  - *Bluetooth*® Low Energy
  - Serial over UART
  - Serial over USB
  - Thread unicast
  - Thread multicast

There are two different DFU package formats:

- Legacy: Uses a simple structure and no security.
- Modern: Uses Google's protocol buffers for serialization and can be cryptographically signed.

The DFU package format transitioned from legacy to modern in nRF5 SDK v12.0.0. Depending on the SDK version that you are using, select a compatible release of this tool:

- Version 0.5.2 generates legacy firmware packages compatible with nRF5 SDK v11.0.0 and earlier.
- Versions 1.5.0 and later generate modern firmware packages compatible with nRF5 SDK v12.0.0 and later.
- Version 2.2.0 or later is required to generate a bootloader settings page that is compatible with nRF52840.

2 Installing nrfutil

You can install nrfutil from the Python Package Index (PyPI) or you can run or install it from the sources.

In both cases, the following prerequisites must be installed:

- Python 2.7 (2.7.6 or later, but not Python 3)
- pip (see Installing Python Modules)

2.1 Installing from PyPI

nrfutil is available as a package in the Python Package Index (PyPI) and can be downloaded and installed directly using the Python installer program **pip**.

Enter the following command to install the latest published version from PyPI:

```
pip install nrfutil
```

This command installs nrfutil and all required packages.

When installing on macOS, you might get an error about the Python module **six**. In this case, enter the following command instead:

```
pip install --ignore-installed six nrfutil
```

If you are running nrfutil on Windows, the runtime libraries targeted during the library build must be present when running code using the library. The following errors indicate that the runtime libraries are not available:

- Missing `MSVC*120.DLL` or `MSVC*140.DLL`
- `RuntimeError: Could not load shared library <path>/pc_ble_driver_shared.dll : '[Error 193] %1 is not a valid Win32 application`

In this case, install the redistributable installer for Visual Studio 2013 or Visual Studio 2015, respectively. Select the version that corresponds to the architecture of your Python installation (x86 or x64).

2.2 Installing from sources

To be able to modify nrfutil, download the sources from GitHub and install the tool.

In addition to Python and **pip**, installing nrfutil from the sources requires the Python setuptools. To upgrade to the latest version, run:

```
pip install -U setuptools
```

If you want to create an executable for nrfutil, install pyinstaller:

```
pip install pyinstaller
```

Complete the following steps to install nrfutil from the sources:

1. Clone the nrfutil GitHub repository.
2. Open a command prompt in the folder where you cloned the repository and run `pip install -r requirements.txt` to install all prerequisites.

3. Set up nrfutil in one of the following ways:

   - Run nrfutil from the sources without installation:

     ```
     python nordicsemi/__main__.py
     ```

     The remainder of this document assumes that you have installed the tool and can call it with `nrfutil`. If you choose to run it without installation, always replace the `nrfutil` command with `python nordicsemi/__main__.py` and add the required command-line options.
   - Install the library to the local Python site-packages and script folder:

     ```
     python setup.py install
     ```

   - Generate a self-contained executable version of the utility:

     ```
     pyinstaller nrfutil.spec
     ```

     Important: Some anti-virus programs will stop **`pyinstaller`** from executing correctly when it modifies the executable file. In this case, configure your anti-virus program to ignore **`pyinstaller`**.

If you are running nrfutil on Windows, the runtime libraries targeted during the library build must be present when running code using the library. The following errors indicate that the runtime libraries are not available:

- Missing `MSVC*120.DLL` or `MSVC*140.DLL`
- `RuntimeError: Could not load shared library <path>/pc_ble_driver_shared.dll : '[Error 193] %1 is not a valid Win32 application`

In this case, install the redistributable installer for Visual Studio 2013 or Visual Studio 2015, respectively. Select the version that corresponds to the architecture of your Python installation (x86 or x64).

3 Displaying help

Add `--help` to any nrfutil command to display help about the command.

Help is context-sensitive. Enter `nrfutil --help` to get information about the general usage of nrfutil, or `nrfutil` *`command`* `--help` to display help for a specific *command*.

For example, enter the following command to display help about executing a DFU procedure over BLE:

```
nrfutil dfu ble --help
```

4 Generating DFU packages

The **pkg** command generates a package to use for a Device Firmware Update. The package contains the new firmware image, an init packet, and a manifest file that indicates the package format. The command can also be used to display the package contents.

Run **nrfutil pkg generate** to generate a zip file that you can use later with a mobile application or another tool to update the firmware of an nRF5 IC. There are several options available, which you can view by entering the following command:

```
nrfutil pkg generate --help
```

Run **nrfutil pkg display** to display the contents of a package.

For example, enter the following command to generate an unsigned package called `app_dfu_package.zip` from the application file `app.hex`:

```
nrfutil pkg generate --application app.hex app_dfu_package.zip
```

Enter the following command to generate a package called `app_dfu_package.zip` from the application file `app.hex` with application version 4 that requires hardware version 51 and SoftDevice S130 v2.0.0 (0x80) and is signed with the private key that is stored in `key.pem`:

```
nrfutil pkg generate --hw-version 51 --sd-req 0x80 --application-version 4
 --application app.hex --key-file key.pem app_dfu_package.zip
```

Enter the following command to generate an unsigned debug package without version information from the application file `app.hex`:

```
nrfutil pkg generate --debug-mode --application app.hex app_dfu_package.zip
```

Enter the following command to display the contents of the created package:

```
nrfutil pkg display app_dfu_package.zip
```

5 Performing a DFU

The **dfu** command performs a Device Firmware Update.

Use this command to perform a device firmware update by transferring a DFU package to the nRF5 IC. Several transports are available. You can display a list of supported transports by entering the following command:

```
nrfutil dfu --help
```

Make sure that the transport you select matches the DFU bootloader that is installed on the DFU target device.

5.1 Performing a DFU over BLE

When performing a Device Firmware Update over a BLE connection, an nRF5 Development Kit board is used as the connectivity IC.

Before performing a DFU over a BLE connection, you must set up your boards to communicate with your computer (see Hardware setup).

Complete the following steps to perform the DFU:

1. Connect an nRF5 Development Kit board to your computer.

   This board is the connectivity IC for performing the DFU on the target. Note that in the `-ic` option, you must specify if the connectivity IC contains an nRF51 or nRF52 chip.
2. Run **`nrfutil dfu ble`** to perform a full DFU procedure over a BLE connection.

   There are several options available, which you can view by entering the following command:

   ```
   nrfutil dfu ble --help
   ```

   Be aware that the `-f` option instructs nrfutil to program the connectivity IC with the required connectivity software. If you specify this option, the contents of the board are overwritten.

   For example, enter the following command to perform a DFU procedure of the file `app_dfu_package.zip` over BLE, using an nRF52 connectivity IC connected to COM3, where the remote BLE device to be upgraded is called MyDevice:

   ```
   nrfutil dfu ble -ic NRF52 -pkg app_dfu_package.zip -p COM3 -n "MyDevice" -f
   ```

5.2 Performing a DFU over Thread

For Device Firmware Updates over Thread, nrfutil supports both unicast and multicast mode.

Before performing a DFU over a Thread connection, you must set up your boards to communicate with your computer (see Hardware setup).

There are two DFU modes in Thread DFU: *unicast* mode and *multicast* mode. In unicast mode, every DFU client requests consecutive blocks of firmware from nrfutil individually. In multicast mode, nrfutil sends consecutive blocks of firmware in multicast messages, and clients that are interested in a new firmware receive and process these messages.

The default mode is unicast mode. To select multicast mode, call nrfutil with a multicast address as a destination address.

Complete the following steps to perform the DFU:

1. Connect an nRF5 Development Kit board to your computer.

   This board serves as the Thread network co-processor (NCP) for performing the DFU on the target.
2. Run **`nrfutil dfu thread`** to perform a full DFU procedure over a Thread connection.

   There are several options available, which you can view by entering the following command:

   ```
   nrfutil dfu thread --help
   ```

   Be aware that the `-f` option instructs nrfutil to program the NCP with the required NCP software. If you specify this option, the contents of the board are overwritten.

   For example, enter the following command to initiate a unicast DFU procedure for the file `app_dfu_package.zip` over Thread on channel 11 with PAN ID 0xABCD, using an nRF52840 NCP connected to COM3:

   ```
   nrfutil dfu thread -f -pkg app_dfu_package.zip -p COM3 --channel 11 --panid
    43981
   ```

   Any remote Thread device can then request the firmware update.

   Enter the following command to perform a multicast DFU procedure for the file `app_dfu_package.zip` over Thread on channel 11 with PAN ID 0xABCD to the multicast address FF03::1, using an nRF52840 NCP connected to COM3:

   ```
   nrfutil dfu thread -f -pkg app_dfu_package.zip -p COM3 --channel 11 --panid
    43981 -r 4 -rs 5000 -a FF03::1
   ```

   Any remote Thread device can then decide whether it wants to receive and process the firmware update messages.

5.3 Performing a DFU over a serial UART connection

The nrfutil tool supports Device Firmware Updates over a serial UART connection.

Complete the following steps to perform the DFU:

1. Connect the DFU target to your computer.

   Most Nordic Semiconductor development boards have an interface MCU that serves as a virtual COM port and transparently maps the UART into a USB CDC ACM interface. See Virtual COM port for more information. If no interface MCU is available, use other means to connect the DFU target to your computer, such as a USB to TTL adapter or a serial cable with an RS-232 connector.
2. Run **`nrfutil dfu serial`** to perform a full DFU procedure over a serial UART connection.

   There are several options available, which you can view by entering the following command:

   ```
   nrfutil dfu serial --help
   ```

For example, enter the following command to perform a DFU procedure of the file `app_dfu_package.zip` over COM3 at 115200 bits per second:

```
nrfutil dfu serial -pkg app_dfu_package.zip -p COM3 -b 115200
```

5.4 Performing a DFU over a serial USB connection

The nrfutil tool supports Device Firmware Updates over a USB CDC ACM connection.

DFU over a serial USB connection is supported only for chips that have USB pins (for example, nRF52840).

Note: The USB port for the interface MCU is not connected to the USB pins on the chip. If you are using the interface MCU, you must perform a DFU over a serial UART connection.

Complete the following steps to perform the DFU:

1. Connect the DFU target to your computer.

   If your DFU target is an nRF52840 Development Kit, use the USB port marked **nRF USB**.
2. Run **`nrfutil dfu usb_serial`** to perform a full DFU procedure over a USB CDC ACM connection.

   There are several options available, which you can view by entering the following command:

   ```
   nrfutil dfu usb_serial --help
   ```

   For example, enter the following command to perform a DFU procedure of the file `app_dfu_package.zip` over COM3 at 115200 bits per second:

   ```
   nrfutil dfu usb_serial -pkg app_dfu_package.zip -p COM3 -b 115200
   ```

6 Generating and displaying keys

The **`keys`** command can be used to generate and display cryptographic keys.

Cryptographic keys are required to sign and validate a Device Firmware Update package. See the Cryptography library in the SDK for more information about signing and cryptographic keys.

- Run **`nrfutil keys generate`** to generate a private (signing) key and store it in a file in PEM format.
- Run **`nrfutil keys display`** to display a private (signing) or public (verification) key from a PEM file.

  There are several options available, which you can view by entering the following command:

  ```
  nrfutil keys display --help
  ```

  For example, enter the following command to generate a private key and store it in a file named `private.pem`:

  ```
  nrfutil keys generate private.pem
  ```

  Enter the following command to display a public key in code format from this key file:

  ```
  nrfutil keys display --key pk --format code private.pem
  ```

7 Generating and displaying bootloader settings

The **settings** command can be used to generate and display a bootloader settings page.

A DFU bootloader requires a bootloader settings page that contains information about the current DFU process. In addition, it can contain information about the installed application and the firmware version.

- Run **nrfutil settings generate** to generate a bootloader settings page and store it in a HEX file.

  There are several options available, which you can view by entering the following command:

  ```
  nrfutil settings generate --help
  ```

- Run **nrfutil settings display** to display the contents of a bootloader settings page that is present in a HEX file.

  To read the bootloader settings page from a programmed IC, use **nrfjprog** to dump the flash memory of the IC (where *HEX_file* is the name of the resulting HEX file):

  ```
  nrfjprog --readcode HEX_file
  ```

After generating the bootloader settings page, you can use **mergehex** and **nrfjprog** to program it to the device. See the nRF5x Command Line Tools documentation for more information.

For example, enter the following command to generate a bootloader settings page for an nRF52840 device with the application `app.hex` installed, with application version 3, bootloader version 2, and bootloader settings version 1 (for SDK v13.0.0), and store it in a file named `settings.hex`:

```
nrfutil settings generate --family NRF52840 --application app.hex --
application-version 3 --bootloader-version 2 --bl-settings-version 1
 settings.hex
```

Enter the following command to display the contents of the generated HEX file:

```
nrfutil settings display settings.hex
```

8 Displaying version information

The **`version`** command can be used to display the version of the tool.

Different versions of nrfutil support different formats of the init packet that is part of the DFU package. Use the nrfutil version that corresponds to the DFU bootloader that is programmed on your device.

Enter the following command to display the nrfutil version:

```
nrfutil version
```

9 Customizing the init packet

If you have created a custom DFU bootloader that uses an init packet of a format different than the one used in the BLE Secure DFU Bootloader, you can customize nrfutil to use this different init packet format.

To customize the tool, you must have cloned the nrfutil GitHub repository (see Installing from sources on page 5).

The init packet is a packet that is sent before the actual firmware images in a Device Firmware Update. It contains metadata about the DFU, such as the size and type of the image, version information, and compability requirements.

If you use the default packet format as described in the BLE Secure DFU Bootloader example, you do not need to modify nrfutil. If you define a custom init packet format, however, you must modify both your DFU bootloader implementation and nrfutil to use this new format.

The format of the init packet is defined in a Protocol buffers (.proto) file. This file can be compiled into different formats, ensuring that you use the same init packet format in you DFU bootloader implementation and in nrfutil.

Important: The init packet definition requires the *proto2* version of the protocol buffers language. Do not include `syntax = "proto3";` in your protocol buffer file, because this would cause the file to be interpreted as a *proto3* language version file.

Complete the following steps to customize the init packet:

1. Modify the protocol buffer file to suit your needs.

   In the nrfutil GitHub repository, the file is located at `nordicsemi/dfu/dfu-cc.proto`. In the BLE Secure DFU Bootloader example in the nRF5 SDK, it is located at `examples/dfu/bootloader_secure/dfu-cc.proto`. Ensure that both files have the exact same content.
2. Download and install the Protocol compiler from Google.
3. Adapt nrfutil to use the new init packet format:
   a) Compile the protocol buffer file to generate the corresponding Python file.

      In the folder that contains your `dfu-cc.proto` file, enter the following command (where *dest_folder* is an empty folder where the protocol compiler will write its output):

      ```
      protoc --python_out=dest_folder dfu-cc.proto
      ```

   b) Copy or move the created file *dest_folder*`/dfu_cc_pb2.py` to `nordicsemi/dfu/`, overwriting the existing file.
   c) If you added new information to the init packet: Adapt nrfutil to include command-line options for new fields and add the information provided through these options to the init packet.

      To adapt the tool, you must edit the Python source files. The contents of the init packet are defined in the files `nordicsemi/dfu/init_packet_pb.py` and `nordicsemi/dfu/package.py`. The command-line options are defined in `nordicsemi/__main__.py`.

   If you installed nrfutil to the local Python site-packages and script folder or created a self-contained executable, you must repeat that procedure after editing the source files.
4. Adapt your DFU bootloader implementation to use the new init packet format:
   a) Make sure that the `dfu-cc.proto` file in the request handling folder of your DFU bootloader implementation (by default, this is the `examples/dfu/dfu_req_handling/` folder in the SDK) is the same file that you used to adapt nrfutil.

b) In the DFU bootloader implementation folder, enter the following command:

```
protoc -odfu-cc.pb dfu-cc.proto
```

This command creates the file `dfu-cc.pb`.

c) If you are not working in the `examples/dfu/dfu_req_handling/` folder, copy the `dfu-cc.options` file from that folder to your implementation folder.

d) Run the `nanopb_generator.py` script (located in the `external/nano-pb/generator/` folder) to generate the required `dfu-cc.pb.c` and `dfu-cc.pb.h` files.

If you are working in `examples/dfu/dfu_req_handling/`, enter the following command:

```
python ../../../external/nano-pb/generator/nanopb_generator.py dfu-cc.pb -f dfu-cc.options
```

If you are working in a different folder, adapt the path to the script.

e) Compile your DFU bootloader.

For more information about this procedure, see the readme file in the BLE Secure DFU Bootloader folder in the SDK.

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.

All information contained in this document represents information on the product at the time of publication. Nordic Semiconductor ASA reserves the right to make corrections, enhancements, and other changes to this document without notice. While Nordic Semiconductor ASA has used reasonable care in preparing the information included in this document, it may contain technical or other inaccuracies, omissions and typographical errors. Nordic Semiconductor ASA assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

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

Nordic Semiconductor products meet the requirements of *Directive 2011/65/EU of the European Parliament and of the Council* on the Restriction of Hazardous Substances (RoHS 2) and the requirements of the *REACH* regulation (EC 1907/2006) on Registration, Evaluation, Authorization and Restriction of Chemicals.

The SVHC (Substances of Very High Concern) candidate list is continually being updated. 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