

# Exercise 1

v2.9.0 – v2.8.0

v2.7.0 – v2.6.1

v2.6.0 – v2.5.0

## Connecting to Wi-Fi using the Wi-Fi shell

In this exercise, we will flash the [Wi-Fi: Shell](#) sample to your board. This sample allows you to send commands to the Wi-Fi chipset on your board through a command-line interface (see [Supported CLI commands](#)).

We will use the CLI to do various common Wi-Fi operations, such as a Wi-Fi scan, connection and disconnection. Afterwards, we will learn how to securely store your Wi-Fi credentials on your device. In the last section, we will test out the commands for managing Target Wake Time (TWT) on your device. This requires you to have a Wi-Fi 6 Access Point with TWT enabled, and this portion of the exercise is, therefore, optional.

### Exercise steps

#### 1. Build and flash the Wi-Fi Shell sample to your board.

- 1.1 In the nRF Connect window in Visual Studio Code, select **Create a new application** under the **WELCOME** menu.
- 1.2 Select **Copy a sample**, then select the nRF Connect SDK version you are using. Search for “Wi-Fi shell” and select the Wi-Fi shell sample with path `nrf/samples/wifi/shell`. In the next step, you will the path to where you want to store this exercise, for example `C:\nordic\l2_e1`.
- 1.3 [Add a build configuration](#) and select the board target for whichever board you are using.

nRF7002 DK

nRF7002 EK

Under **Board**, select `nrf7002dk_nrf5340_cpuapp`, then click **Build Configuration**.



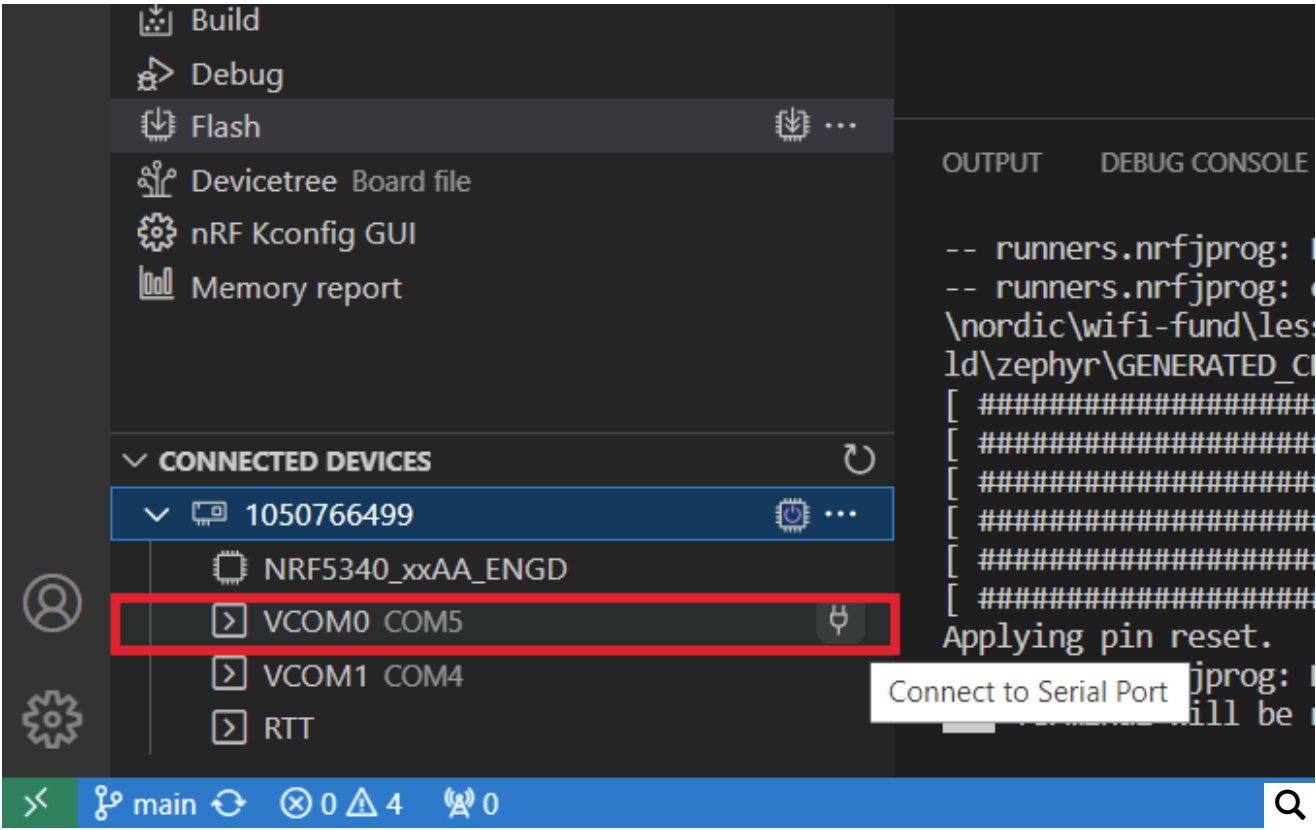
Take Notes





×

Completed



Observe the following log output

```
OK
[00:00:00.620,391] <inf> fs_nvs: 6 Sectors of 4096 bytes
[00:00:00.620,422] <inf> fs_nvs: alloc wra: 0, fe8
[00:00:00.620,422] <inf> fs_nvs: data wra: 0, 0
*** Booting nRF Connect SDK 2.6.1-3758bcbfa5cd ***
Starting nrf7002dk_nrf5340_cpuapp with CPU frequency: 128 MHz
[00:00:00.620,758] <inf> wpa_supp: Successfully initialized wpa_supplicant
```

2. Scan for access points in your vicinity.

Before we connect to a network, let's do a passive scan to see all available access points.

Type the following command into the terminal.

```
wifi scan
```

You should see something similar to the following output (this might take a few seconds). This is the scan result list.

Num	SSID	(len)	Chan	(Band)	RSSI	Security	BSSID	MFP
1	Wi-Fi-Network-Guest	12	11	(2.4GHz)	-74	WPA2-PSK	10:05:CA:56:89:E0	Disable
2	Wi-Fi-Network-Internal	15	11	(2.4GHz)	-74	EAP	CC:16:7E:15:04:A1	Disable
3	Wi-Fi-net	7	6	(2.4GHz)	-77	WPA-PSK	FC:34:97:0B:F6:D8	Disable
4	Wi-Fi-Network-Guest	12	64	(5GHz )	-79	WPA2-PSK	10:05:CA:56:89:EF	Disable
5	Wi-Fi-Network-Internal	15	64	(5GHz )	-81	EAP	CC:16:7E:15:04:AE	Disabl

The Wi-Fi chip outputs the acquired information about each available access point. As you can see in the output, this information includes the SSID, the length of the SSID, the channel where the AP was broadcasting its beacon, the band the AP operates on, the RSSI as perceived by your device, the security protocol implemented by the AP, the BSSID, and finally the presence or absence of MFP security.

3. Connect to an access point.

Now we can use the command `wifi connect` to connect to an access point.

`wifi connect` takes the following parameters

- s <SSID>: SSID of network
- c <channel>: 0 for any channel
- b <band>: 0 for any band
- p <PSK>: Valid only for secured SSIDs
- k <key-mgmt>: 0-None, 1-WPA2-PSK, 2-WPA2-PSK-256, 3-SAE, 4-WAPI, 5-EAP, 6-WEP,



- -m <bssid>: MAC address of the AP (BSSID).

This is a Unsubmitted Revision.

[View Queue](#)

[Compare](#)

[View Published Post](#)

[Edit](#)

[Submit](#)

[Publish now](#)

- -h <help>: Print out the help for the connect command.

Entering only the SSID, password (PSK), and key management of the access point you want to connect to will suffice. You can use the results from the `wifi scan` command to know which key management parameter to enter, see the Security column.

For the <key-mgmt> parameter, entering “0” only works with open APs.

Issue the following command in the terminal to connect to the AP.

```
wifi connect -s "<SSID>" -p "<psk>" -k <key-mgmt>
```

If the connection is successful, you should see a similar log output

```
wifi connect -s "<SSID>" -p "<psk>" -k <key-mgmt>
wifi connect -s "<SSID>" -p "<psk>" -k <key-mgmt>
OK
OK
OK
OK
OK
OK
Connection requested
Connected
wpa_supp: wlan0: SME: Trying to authenticate with aa:aa:aa:aa:aa:aa (SSID='<SSID>' freq=5180 MHz)
<inf> wifi_nrf: wifi_nrf_wpa_supp_authenticate: Authentication request sent successfully
<inf> wpa_supp: wlan0: Trying to associate with aa:aa:aa:aa:aa:aa (SSID='<SSID>' freq=5180 MHz)
<inf> wifi_nrf: wifi_nrf_wpa_supp_associate: Association request sent successfully
<inf> wpa_supp: wlan0: Associated with aa:aa:aa:aa:aa:aa
<inf> wpa_supp: wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
<inf> wpa_supp: wlan0: WPA: Key negotiation completed with aa:aa:aa:aa:aa:aa [PTK=CCMP GTK=CCMP]
<inf> wpa_supp: wlan0: CTRL-EVENT-CONNECTED - Connection to aa:aa:aa:aa:aa:aa completed [id=0]
<inf> net_dhcpv4: Received: 192.00.00.00
```

#### Note



The multiple “OK” messages come from the Wi-Fi stack and will be omitted in later log output examples for clarity.

#### 4. Get the status of the network interface.

The command `wifi status` lets us read out extra information about the access point we are connected to that we didn't get during the scan procedure, such as the beacon interval, DTIM and whether TWT is supported or not.

Issue the following command in the terminal

```
wifi status
```

Observe the following output:

```
Status: successful
=====
State: COMPLETED
Interface Mode: STATION
Link Mode: WIFI 6 (802.11ax/HE)
SSID: <SSID>
BSSID: AA:AA:AA:AA:AA:AA
Band: 5GHz
Channel: 36
Security: WPA2-PSK
MFP: Optional
RSSI: -86
```

0%

Completed

Take Notes





## 5. Get statistics about the Wi-Fi interface.

This is a Unsubmitted Revision.

[View Queue](#)[Compare](#)[View Published Post](#)[Edit](#)[Submit](#)[Publish now](#)

Issue the following command in the terminal

```
wifi statistics
```

Observe the following output

```
Statistics for Wi-Fi interface 0x200016a8 [1]
Bytes received   : 109247
Bytes sent       : 5458
Packets received : 346
Packets sent     : 39
Receive errors   : 0
Send errors      : 0
Bcast received   : 371
Bcast sent       : 11
Mcast received   : 218
Mcast sent       : 8
Beacons received : 138
Beacons missed   : 46
```

## 6. Configure device wakeup mode

The command `wifi ps_wakeup_mode` is used to configure your Wi-Fi device wakeup mode when using power save mode. It takes the following parameters.

- `<dtim>`: Wakeup mode for the DTIM interval
- `<listen_interval>`: Wakeup mode for the Listen interval

Issue the following command in the terminal to set it to wakeup for the DTIM interval for example. Power save modes, DTIM intervals and the Listen interval will be explained in detail in Lesson 6 of this course.

```
wifi ps_wakeup_mode dtim
```

## 7. Storing the Wi-Fi credentials.

Using the `wifi connect` command mentioned above, you will have to re-enter your SSID and password upon device reset to re-establish the connection. To mitigate this, we can use the Wi-Fi credentials shell subcommands, which interact with the Wi-Fi credentials library to add the network credentials to credentials storage.

7.1 Store the Wi-Fi credentials using `wifi_cred add`.

To store the credentials, we will be using the Wi-Fi credentials shell, which are identified as `wifi_cred`, and the subcommand `add`.

We will be using the following parameters from `wifi_cred add`

- `<-s -ssid "<SSID>">`: SSID.
- `[-p, -passphrase]`: Passphrase (valid only for secure SSIDs)
- `[-k, -key-mgmt]`: Key management type: 0: None, 1: WPA2-PSK, 2: WPA2-PSK-256, 3: SAE-HNP, 4: SAE-H2E, 5: SAE-AUTO, 6: WAPI, 7: EAP-TLS, 8: WEP, 9: WPA-PSK, 10: WPA-Auto-Personal, 11: DPP

Enter the following CLI command to store your Wi-Fi credentials

```
wifi_cred add -s "<your_network_SSID>" -p "<your_network_password>" -k <key_mgmt>
```

For the last command, refer to the output from `wifi scan`, specifically the security column. Then replace `<key-mgmt>` with whichever number corresponds to the security for the Wi-Fi network you wish to connect to.

You can check them out by using this command

0%

Completed

Take Notes



```
wifi_cred auto_connect
```

The auto\_connect command is defined in `<install_path>/nrf/subsys/net/lib/wifi_mgmt_ext/wifi_cred_shell_autoconnect.c`, and uses the NET\_REQUEST\_WIFI\_CONNECT\_STORED command in the Network Management API. We will take a closer look at this command in Exercise 2 of this lesson.

7.3 The wifi\_cred command also has the subcommands list and delete.

list will list all the networks in credential storage and delete "<SSID>" will remove that network from credentials storage.

```
wifi_cred list
wifi_cred delete "<SSID>"
```

### (Optional) 8. Set up a Target Wake Time flow.

This step is optional as it only works if you are using a Wi-Fi 6 router that supports Target Wake Time. In some cases, you will need to enable TWT functionality in the router configurations. Consult your router's user manual for more information.

The command wifi twt is used to manage TWT flows.

It has three different subcommands: setup, teardown, and teardown\_all.

To setup a TWT flow, we will use the setup subcommand, which takes the following parameters

- <negotiation\_type>: 0 — Individual, 1 — Broadcast, 2 — Wake TBTT
- <setup\_cmd>: 0 — Request, 1 — Suggest, 2 — Demand
- <dialog\_token>: 1-255, used to match action requests with action responses
- <flow\_id>: 0-7, gives an ID to the flow being created
- <responder>: 0- Requester, 1- Responder
- <trigger>: 0- Trigger mode enabled, 1- Trigger mode disabled
- <implicit>: 0- Explicit TWT operation, 1- Implicit
- <announce>: 0- Announce mode enabled, 1- Announce mode disabled
- <twt\_wake\_interval>: 1 — 262144  $\mu$ s, TWT Wake Duration, for how long will the device be awake (in  $\mu$ s)
- <twt\_interval>: 1  $\mu$ s —  $2^{31}$   $\mu$ s, TWT Wake Interval, the interval between successive TWT wake periods

Negotiation type is used to determine the target STA of this TWT schedule. When set to "Individual", it means that this TWT schedule will be set to the STA individually. Whereas, when set to "Broadcast", it means the AP broadcasts TWT schedules to multiple STAs.

We want the device to wakeup for 20 ms, every minute.

Issue the following command in the terminal

```
wifi twt setup 0 0 2 2 0 1 1 1 20000 60000000
```

### (Optional) 9. Tear down the Target Wake Time flow.

To tear down the TWT flow, we will use the teardown subcommand, which takes the following parameters

- <negotiation\_type>: 0 — Individual, 1 — Broadcast, 2 — Wake TBTT
- <setup\_cmd>: 0 — Request, 1 — Suggest, 2 — Demand
- <dialog\_token>: 1-255, used to match action requests with action responses
- <flow\_id>: 0-7, gives an ID to the flow being created

Use the flow\_id parameter to indicate which TWT flow to tear down.

We want to tear down the flow that was set up in the previous step, which we assigned flow\_id 2.



wifi twt teardown 0 0 2 2

This is a Unsubmitted Revision.

[View Queue](#)

[Compare](#)

[View Published Post](#)

[Edit](#)

[Submit](#)

[Publish now](#)

Alternatively, teardown\_all can be used to tear down all TWT flows, by sending the following command.

```
wifi twt teardown_all
```



Take Notes



Completed

0%