

- DFU OTA app with Secure bootloader (sdk 12.3.0) on nrf51 via Android phone
 - Values sent-received
 - Explanation
 - * Information from sdk documentation
 - * Applying
 - Reverse app.dat
 - * Some examples
 - * Results
 - How to compute CRC32:

DFU OTA app with Secure bootloader (sdk 12.3.0) on nrf51 via Android phone

Values sent-received

Handle	Sent	Received
0x000e	0x0001	
0x000d	0601	600601 0001 0000 0000 0000
		0000 0000
	02 0000	600201
	0101 8d00	600101
0x000b	128a010a4408011240080110331a028701200028 00300038f4fa0242240803122028229c1855b166 741e44ab8a6eac2847cebd6ced2d2f4101276bf0 fa2b7befc6480052040801120010001a408e9277 297e36ee11d255c8e5a451e3a4bdd3359ab060d6 36185f93f8a521097b0ea3f384dd01c9845d9839 88d2775e0079a99a3c50e377435cbf426ea10f77 14	
0x000d	03	600301 8d00 0000 c5 6c 4d 24
	04	600401
	02 0a00	600201
	0602	600601 0010 0000 0000 0000
		0000 0000
	0102 0010 0000	600101
0x000b	0080002021590200615902006359020000000000 00 0000000065590200000000000000000067590200 69590200f1df01006b59020021e901006b590200 6b5902000000000041e401006b5902006b590200 6b5902006b5902006b5902006b5902006b590200	

Handle Sent	Received
6b5902006b5902006b59020015b701006b590200	
6b590200f1b701006b590200816402006b590200	
6b5902006b590200000000000000000000000000	
00000000000000000000000000448054b10b58342	
	600301 c8000000 82899e87
	...
	...
	600301 90010000 188eb8a5
	600301 58020000 38386f1b
	600301 20030000 b448eecf
	600301 e8030000 b1e2742f
	600301 b0040000 db42e210
	600301 78050000 a04399c6
	600301 40060000 c3d73b9c
	600301 08070000 7fe50ea9
	600301 d0070000 122a20a0
	600301 98080000 898d0f60
	600301 60090000 7b69fdec
	600301 280a0000 9feb7d51
	600301 f00a0000 5639f129
	600301 b80b0000 22068d1e
	600301 800c0000 9ee5c511
	600301 480d0000 4aad509c
	600301 100e0000 2bf3bfea
	600301 d80e0000 ea43e47b
	600301 a00f0000 fe1e226c
03	600301 00100000 69aa84d7
04	600401
0102 0010 0000	600101

Continue to send totally 4096 bytes. | | | | — | ——— | ————— |
| 0x000d | 03 | 6003010010000069aa84d7 | | 04 | 600401 | | 010200100000 |
600101 | | | | Then send 4096 bytes and so on.

Some of the last write commmands are:

	600301 48bd0000
	09642611
00000000181818181818181818181818000000024f400	
35b1010009b10100000000000000000000000001ca0100	
03ff0000	

03	600301 74bd0000 0e6e5c4b
04	600401

Explanation

Information from sdk documentation

https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v12.3.0/1ib_dfu_transport_ble.html

Characteristic name	Required properties	Optional properties	UUID
DFU Control Point	Write, Notify		0x8EC90001-F315-4F60-9FB8-838830DAEA50
DFU Packet	WriteWithoutResponse, Notify		0x8EC90002-F315-4F60-9FB8-838830DAEA50

Figure 1: image

Control point procedure operation codes and the respective parameters and response values:

The result codes that are sent as part of the response:

Applying

- 0001: trigger the DFU
- 0601:
 - 06: Select
 - 01: Create command object
- 60 0601 0001 0000 0000 0000 0000 0000:
 - 60: Response code
 - 06: request opcode
 - 01: result code
 - 0001 0000: Maximum size
 - 0000 0000: Offset
 - 0000 0000: CRC32, initial value
- 02 0000: Set PRN value to be 0
- 600201:
 - 60: Response code
 - 02: request opcode
 - 01: success
- 0101 8d00 0000:
 - 0101: Create command object
 - 8d00 0000: Size in little endian. 0x8d= 141
- 600101:

Opcode	Procedure	Description	Parameters	Response value
0x01	Create	Creates an object with the given type and selects it. Removes an old object of the same type (if such an object exists).	Type (uint8_t), Size (uint32) Types: <ul style="list-style-type: none"> • 0x01 - Command object • 0x02 - Data object (firmware) Size: <ul style="list-style-type: none"> • Object size in little endian 	
0x02	Set Packet Receipt Notification (PRN) value	Sets the number of packets to be sent before receiving a Packet Receipt Notification. The default is 0.	Value (uint16, little endian)	
0x03	Calculate checksum	Requests the checksum of the object that was sent (applicable only for data objects). The checksum is reset after sending an Execute command.	None	Offset (uint32), CRC32 (uint32)
0x04	Execute	Executes the last object that was sent.	None	
0x06	Select	Selects the last object with the given type that was sent.	Type (uint8_t) Types: <ul style="list-style-type: none"> • 0x01 - Command object • 0x02 - Data object (firmware) 	Maximum size (uint32), offset (uint32), CRC32 (uint32)
0x60	Response Code		Request opcode, result code	

Figure 2: image

Result code	Definition	Description
0x00	Invalid code	The provided opcode was missing or malformed.
0x01	Success	The operation completed successfully.
0x02	Opcode not supported	The provided opcode was invalid.
0x03	Invalid parameter	A parameter for the opcode was missing.
0x04	Insufficient resources	There was not enough memory for the data object.
0x05	Invalid object	The data object did not match the firmware and hardware requirements, the signature was missing, or parsing the command failed.
0x07	Unsupported type	The provided object type was not valid for a Create or Read operation.
0x08	Operation not permitted	The state of the DFU process did not allow this operation.
0x0A	Operation failed	The operation failed.

Figure 3: image

- 60: response code
- 01: request code
- 01: success

Then the client sends data to the GATT server. Each package maximum size is 20 bytes which is (MTU-3). There are 7 packages of 20 bytes and one package of 1 byte. Totaly, there are 141 bytes.

- 03: request calculate checksum
- 600301 8d000000 c56c4d24:
 - 60: response code
 - 03: request code
 - 01: success
 - 8d000000: offset
 - c5 6c 4d 24: CRC32
- 04: Execute the last object that was sent
- 02 0a00:
 - 02: Set PRN value
 - 0a00: little endian, which is 10. It means that client will receive a notification every 10 “write” commands.
- 600201:
 - 60: response code
 - 02: request code
 - 01: success
- 0602:
 - 06: select last object
 - 02: data object (firmware)
- 600601 0010 0000 0000 0000 0000 0000:
 - 60: response code
 - 06: request code
 - 01: success
 - 0010 0000: maximumsize
 - 0000 0000: offset
 - 0000 0000: CRC32
- 0102 0010 0000:
 - 01: Create
 - 02: Data object
 - 0010 0000: Size in little endian: 4096
- 600101:
 - 60: response code
 - 01: request code
 - 01: success

Reverse app.dat

Some examples

Softdevices 0x1234 - application version 1:

```
-Over-the-Air-firmware-updates/python-code/dfu_app51-sd0x1234v1]
$ hexdump app.dat
00000000 8a12 0a01 0844 1201 0840 1001 1a33 b402
0000010 2024 2800 3000 3800 faf4 4202 0824 1203
0000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
0000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
0000040 48c6 5200 0804 1201 1000 1a00 b540 9ea3
0000050 fe36 09fa 9d93 6e74 d30f 1448 42d3 6414
0000060 ce38 d186 06ba 21a4 1695 3039 1adf 298c
0000070 7bba c97d 74a4 35db 2dd2 d7b5 1b3f 34bc
0000080 18dc acf1 fe41 32df aca1 526e 0083
000008d
```

Figure 4: image

Softdevices 0x1111 - application version 1:

```
-Over-the-Air-firmware-updates/python-code/dfu_app51-sd0x1111v1]
$ hexdump app.dat
00000000 8a12 0a01 0844 1201 0840 1001 1a33 9102
0000010 2022 2800 3000 3800 faf4 4202 0824 1203
0000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
0000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
0000040 48c6 5200 0804 1201 1000 1a00 8240 7d06
0000050 601b 308c 5472 b111 44d1 bb5d dde9 8a6d
0000060 e72b 56c3 ab32 7fc3 c08e ae12 f8de 9ed7
0000070 1f18 5337 5778 cfd8 2ffb 8867 db19 7b47
0000080 6a07 9118 8251 d202 823b fc4f 005b
000008d
```

Figure 5: image

Softdevices 0x87 - application version 9:

Hardware version 52

Softdevices 0x1234 - application version 1:

Results

- 1: application version
- 2: softdevice version
- 3+4: hardware version
- 5: signature of the hash

```

-Over-the-Air-firmware-updates/python-code/dfu_app51-sd-x87v9]
└─$ hexdump app.dat
00000000 8a12 0a01 0844 1201 0840 1009 1a33 8702
00000010 2001 2800 3000 3800 faf4 4202 0824 1203
00000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
00000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
00000040 48c6 5200 0804 1201 1000 1a00 a340 8f08
00000050 0ea8 879e 967e 84eb e157 6b10 e14a d220
00000060 2694 06bd 308b bc0e 4dd1 8ef5 5b15 3209
00000070 90a4 46c0 ff8f 6bfb 9713 e4ad 77d1 36f9
00000080 76cb 2618 3be1 fae0 89b9 33b6 0037
0000008d

```

Figure 6: image

```

-Over-the-Air-firmware-updates/python-code/dfu_app52-sd0x1234v1]
└─$ hexdump app.dat
00000000 8a12 0a01 0844 1201 0840 1001 1a34 b402
00000010 2024 2800 3000 3800 faf4 4202 0824 1203
00000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
00000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
00000040 48c6 5200 0804 1201 1000 1a00 4940 81ec
00000050 62c2 bc2c 4cd6 1c11 5fba d9bd 72eb bae9
00000060 72cd d4e7 5696 fd9f 40ec 6dff 62b1 0eb9
00000070 8e05 e74d 0493 86b7 6790 49c6 1bea 8a35
00000080 ff70 a3fa f1c4 799f 7692 045d 00a8
0000008d

```

Figure 7: image

```

└─$ hexdump app.dat
00000000 8a12 0a01 0844 1201 0840 1001 1a33 8702
00000010 2001 2800 3000 3800 faf4 4202 0824 1203
00000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
00000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
00000040 48c6 5200 0804 1201 1000 1a00 8e40 7792
00000050 7e29 ee36 d211 c855 a4e5 e351 bda4 35d3
00000060 b09a d660 1836 935f a5f8 0921 0e7b f3a3
00000070 dd84 c901 5d84 3998 d288 5e77 7900 9aa9
00000080 503c 77e3 5c43 42bf a16e 770f 0014
0000008d

```

Figure 8: image

Not sure if these values are two bytes or one byte long.

How to compute CRC32:

nRF5 SDK 12.3.0 \components\libraries\crc32\crc32.c