# Modifying the nRF8001 Setup data

Contents

[Modifying the nRF8001 Setup data 1](#_Toc388440711)

[Configuration Types 2](#_Toc388440712)

[\* @enum aci\_cmd\_priv\_setup\_target\_t 2](#_Toc388440713)

[Overall format of the nRF8001 Setup area 2](#_Toc388440714)

[\* @struct dm\_mrs\_data\_t 2](#_Toc388440715)

[\* @struct dm\_ds\_setup\_signature\_t 3](#_Toc388440716)

[VERSIONS 0x0 3](#_Toc388440717)

[\* @struct aci\_cmd\_priv\_setup\_params\_versions\_t 3](#_Toc388440718)

[Device Settings - DEVSETT 0x1 4](#_Toc388440719)

[\* @struct dm\_dev\_settings\_t 4](#_Toc388440720)

[gatt\_dm\_perm\_t 4](#_Toc388440721)

[\* @struct dm\_setup\_counts\_t 4](#_Toc388440722)

[\* @struct dm\_ds\_hw\_data\_t 4](#_Toc388440723)

[\* @struct dm\_ds\_ad\_data\_t 5](#_Toc388440724)

[\* @struct dm\_ds\_security\_data\_t 6](#_Toc388440725)

[\* @struct dm\_ds\_v2\_data\_t 6](#_Toc388440726)

[\* @struct dm\_ds\_v3\_data\_t 7](#_Toc388440727)

[ATTDB 0x2 7](#_Toc388440728)

[attdb\_head\_t 8](#_Toc388440729)

[/\*\* @struct attdb\_record\_head\_t 8](#_Toc388440730)

[SERVICES 0x3 9](#_Toc388440731)

[\* @struct dm\_service\_t 9](#_Toc388440732)

[Attribute abstraction - ATTRS 0x4 10](#_Toc388440733)

[\* @struct dm\_attr\_t 10](#_Toc388440734)

[Vendor Specific UUIDS - VSUUIDS 0x5 10](#_Toc388440735)

[uint128\_le\_t 10](#_Toc388440736)

[EXTATTRS 0x6 10](#_Toc388440737)

[\* @struct dm\_ext\_attr\_t 10](#_Toc388440738)

[Custom advertising types - CUSTADTYPES 0x7 11](#_Toc388440739)

[\* @struct custom\_ad\_type\_t 11](#_Toc388440740)

[CRC 11](#_Toc388440741)

[Example ble\_modify\_setup\_data 12](#_Toc388440742)

[CRC used: 12](#_Toc388440743)

[ATTDB 0x20 12](#_Toc388440744)

[attdb\_head\_t 13](#_Toc388440745)

[/\*\* @struct attdb\_record\_head\_t 14](#_Toc388440746)

# Configuration Types

Message format:

Length: 1 byte

Opcode = 0x06, 1 byte

Target = 4 bit

Offset = 12 bits

Eksample: Look for the message <Len> 0x06 0x70 0x00 <Custom advertising data follows>

/\*\*

### \* @enum aci\_cmd\_priv\_setup\_target\_t

\* @brief ACI setup targets

\*/

typedef enum

{

ACI\_SETUP\_TARGET\_VERSIONS = 0x0,

ACI\_SETUP\_TARGET\_DEVSETT = 0x1,

ACI\_SETUP\_TARGET\_ATTDB = 0x2,

ACI\_SETUP\_TARGET\_SERVICES = 0x3,

ACI\_SETUP\_TARGET\_ATTRS = 0x4,

ACI\_SETUP\_TARGET\_VSUUIDS = 0x5,

ACI\_SETUP\_TARGET\_EXTATTRS = 0x6,

ACI\_SETUP\_TARGET\_CUSTADTYPES = 0x7,

ACI\_SETUP\_TARGET\_DEVCFG = 0x8,

ACI\_SETUP\_TARGET\_CRC = 0xF

} aci\_cmd\_priv\_setup\_target\_t;

# Overall format of the nRF8001 Setup area

/\*\*

## \* @struct dm\_mrs\_data\_t

\* @brief Memory Retention Setup Data

\*/

typedef struct

{

dm\_setup\_signature\_t setup\_signature; /\* This is the first byte written/read to/from OTP \*/

dm\_dev\_settings\_t device\_settings;

uint8\_t CLASS\_XDATA \*attdb\_heap;

dm\_service\_t CLASS\_XDATA \*rem\_svcs;

dm\_attr\_t CLASS\_XDATA \*attrs;

uint128\_le\_t CLASS\_XDATA \*vs\_uuids;

dm\_ext\_attr\_t CLASS\_XDATA \*ext\_attrs;

dm\_custom\_ad\_type\_t CLASS\_XDATA \*cust\_ad\_types;

} dm\_mrs\_data\_t;

/\*\*

### \* @struct dm\_ds\_setup\_signature\_t

\* @brief Setup Data signature

\*/

typedef struct

{

uint8\_t setup\_format : 7;

uint8\_t locked : 1;

} dm\_setup\_signature\_t;

# VERSIONS 0x0

/\*\*

### \* @struct aci\_cmd\_priv\_setup\_params\_versions\_t

\* @brief Structure for the ACI\_CMD\_PRIV\_SETUP parameters with target ACI\_SETUP\_TARGET\_VERSIONS

\*/

typedef struct

{

uint8\_t setup\_format;

uint8\_t aci\_version;

uint16\_t host\_build;

} aci\_cmd\_priv\_setup\_params\_versions\_t;

/\*\*

#### \* @def ACI\_SETUP\_FORMAT

\* @brief Current ACI Setup Format supported

\*/

#define ACI\_SETUP\_FORMAT 0x03

/\*\*

#### \* @def ACI\_VERSION

\* @brief Current ACI protocol version. 0 means a device that is not yet released.

\* A numer greater than 0 refers to a specific ACI version documented and released.

\* The ACI consists of the ACI commands, ACI events and error codes.

\*/

#define ACI\_VERSION (0x02)

# Device Settings - DEVSETT 0x1

/\*\*

## \* @struct dm\_dev\_settings\_t

\* @brief Device Settings

\*/

typedef struct

{

uint32\_t setup\_id;

gatt\_dm\_perm\_t db\_perm; /\*\*< Host Security level \*/

dm\_setup\_counts\_t counts;

dm\_ds\_hw\_data\_t hw\_data;

dm\_ds\_ad\_data\_t ad\_data;

dm\_ds\_security\_data\_t sec\_data;

/\* v2 extra data \*/

dm\_ds\_v2\_data\_t v2\_data;

dm\_ds\_v3\_data\_t v3\_data;

} dm\_dev\_settings\_t;

### gatt\_dm\_perm\_t

/\*\* @brief Pipe permissions

\*/

typedef struct {

uint8\_t perm:2;

uint8\_t resv0:6;

} gatt\_dm\_perm\_t;

#define GATT\_DM\_PERM\_OPEN 0x00

#define GATT\_DM\_PERM\_TRUSTED 0x01

#define GATT\_DM\_PERM\_TRUSTED\_ENCRYPTED 0x02

#define GATT\_DM\_PERM\_MASK (GATT\_DM\_PERM\_OPEN | GATT\_DM\_PERM\_TRUSTED | GATT\_DM\_PERM\_TRUSTED\_ENCRYPTED)

/\*\*

### \* @struct dm\_setup\_counts\_t

\* @brief Setup table sizes

\*/

typedef struct

{

uint8\_t rem\_svc\_count;

uint8\_t loc\_attr\_count;

uint8\_t rem\_attr\_count;

uint8\_t pipe\_count;

} dm\_setup\_counts\_t;

/\*\*

### \* @struct dm\_ds\_hw\_data\_t

\* @brief Hardware device settings

\*/

typedef struct

{

dm\_ds\_clock\_parameters\_t clock\_params;

dm\_ds\_active\_signal\_t active\_signal;

uint8\_t dcdc\_conv : 1;

uint8\_t output\_power : 7;

uint8\_t gain;

} dm\_ds\_hw\_data\_t;

/\*\*

#### \* @struct dm\_ds\_clock\_parameters\_t

\* @brief Mirror the LL clock parameters structure

\*/

typedef struct

{

uint8\_t clock\_source\_32khz;

uint8\_t sleep\_clock\_accuracy;

uint8\_t clock\_source\_16mhz;

} dm\_ds\_clock\_parameters\_t;

/\*\*

#### \* @struct dm\_ds\_active\_signal\_t

\* @brief Mirror the LL active signal structure

\*/

typedef struct

{

uint8\_t mode : 2;

uint8\_t signal\_to\_tick\_distance : 6;

} dm\_ds\_active\_signal\_t;

/\*\*

### \* @struct dm\_ds\_ad\_data\_t

\* @brief Advertisement Data device settings

\*/

typedef struct

{

/\* Shortened name length \*/

uint8\_t name\_short\_len;

dm\_ds\_ad\_uuid\_cnt\_t loc\_svcuuid\_cnt;

/\* Local 16-bit SVC UUIDs in Little Endian \*/

uint16\_t loc\_svcuuid\_16[ACI\_AD\_LOC\_SVCUUID\_16\_MAX\_COUNT];

/\* Local 128-bit SVC UUIDs in uuid\_t format \*/

uuid\_t loc\_svcuuid\_128[ACI\_AD\_LOC\_SVCUUID\_128\_MAX\_COUNT];

dm\_ds\_ad\_uuid\_cnt\_t sol\_svcuuid\_cnt;

/\* Solicited 16-bit SVC UUIDs in Little Endian \*/

uint16\_t sol\_svcuuid\_16[ACI\_AD\_SOL\_SVCUUID\_16\_MAX\_COUNT];

/\* Solicited 128-bit SVC UUIDs in uuid\_t format \*/

uuid\_t sol\_svcuuid\_128[ACI\_AD\_SOL\_SVCUUID\_128\_MAX\_COUNT];

/\* AD bitmaps: Big Endian \*/

uint32\_t bm\_bitmap;

uint32\_t gm\_bitmap;

} dm\_ds\_ad\_data\_t;

/\*\*

#### \* @struct dm\_ds\_ad\_uuid\_cnt\_t

\* @brief UUID count info structure (16/128/complete)

\*/

typedef struct

{

uint8\_t uuid\_16\_cnt : 4;

uint8\_t uuid\_128\_cnt : 2;

uint8\_t uuid\_16\_complete : 1;

uint8\_t uuid\_128\_complete : 1;

} dm\_ds\_ad\_uuid\_cnt\_t;

#### Uuid\_t

/\*\* @brief UUID extended type, classifies SIG and vendor-specific UUID values

\*/

typedef struct {

uint16\_t uuid; /\*\*< 16-bit UUID in platform endianness \*/

uint8\_t type; /\*\*< uuid\_type\_t, uint8\_t used for packing, as uuid\_type\_t size is compiler dependent \*/

} uuid\_t;

/\*\*

### \* @struct dm\_ds\_security\_data\_t

\* @brief Security device settings

\*/

typedef struct

{

uint8\_t io\_caps : 3;

uint8\_t oob : 1;

uint8\_t auth\_req : 2;

uint8\_t key\_req : 2;

/\* 0 - 9 (7 - 16)\*/

uint8\_t min\_keysize : 4;

/\* 0 - 9 (7 - 16)\*/

uint8\_t max\_keysize : 4;

} dm\_ds\_security\_data\_t;

/\*\*

### \* @struct dm\_ds\_v2\_data\_t

\* @brief Additional data for setup format 0x02

\*/

typedef struct

{

uint8\_t vs\_uuid\_count;

} dm\_ds\_v2\_data\_t;

/\*\*

### \* @struct dm\_ds\_v3\_data\_t

\* @brief Additional data for setup format 0x03

\*/

typedef struct

{

dm\_ds\_window\_limit\_t window\_limit;

uint16\_t bond\_timeout;

uint8\_t sec\_req\_delay;

uint8\_t l2c\_req\_delay;

struct {

uint32\_t bm\_sr;

uint32\_t gm\_sr;

uint32\_t bcm;

uint32\_t bcm\_sr;

uint8\_t cust\_bm;

uint8\_t cust\_bm\_sr;

uint8\_t cust\_gm;

uint8\_t cust\_gm\_sr;

uint8\_t cust\_bcm;

uint8\_t cust\_bcm\_sr;

} ad\_bitmaps;

struct {

uint8\_t whitelist\_disable : 1; /\* BTHOST-1193 \*/

uint8\_t reserved : 7; /\* RFI \*/

} features;

uint8\_t custom\_ad\_types\_count;

} dm\_ds\_v3\_data\_t;

#### dm\_ds\_window\_limit\_t

typedef struct

{

uint8\_t limit;

uint8\_t dropped\_pkt\_threshold;

uint8\_t auto\_off\_count;

} dm\_ds\_window\_limit\_t;

# ATTDB 0x2

/\*\* @file

\*

\* @brief This header file contains ATT DB internl data types

\* Inside the ATTDB, handles and UUIDs are stored in BE and values in LE

\* Please note that the ATT protocol requires data in LE format

\* ATTDB\_P2D macros store data in BE, for handles and UUIDs

\* uuid\_encode with UUID\_P2U\_UINT16 stores data in LE, for values

\*/

### attdb\_head\_t

/\*\* @brief Attribute permissions and length field

\*/

typedef struct {

/\* octet 0 \*/

uint8\_t mlen1:2; /\*\*< MSb of max len, refer Macros section in attdb.c \*/

uint8\_t valid:1; /\*\*< Valid record \*/

uint8\_t bcast:1; /\*\*< Broadcastable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t notify:1; /\*\*< Notifiable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t ind:1; /\*\*< Indicatable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t flighty:1; /\*\*< Flighty/volatile record \*/

uint8\_t locked:1; /\*\*< Lock (RFI) \*/

/\* octet 1 \*/

uint8\_t vlen1:2; /\*\*< MSb of actual len, refer Macros section in attdb.c \*/

uint8\_t rd:2; /\*\*< Read permisions (none, plaintext, authenticated plaintext, authenticated encrypted) \*/

uint8\_t wr:2; /\*\*< Write permisions (none, plaintext, authenticated plaintext, authenticated encrypted) \*/

uint8\_t ard:1; /\*\*< Read authorization (RFI) \*/

uint8\_t awr:1; /\*\*< Write authorization (RFI) \*/

/\* octet 2 \*/

uint8\_t mlen0:8; /\*\*< LSb of max len, refer Macros section in attdb.c \*/

/\* octet 3 \*/

uint8\_t vlen0:8; /\*\*< LSb of actual len, refer Macros section in attdb.c \*/

} attdb\_head\_t;

### /\*\* @struct attdb\_record\_head\_t

\*

\* @brief Attribute DB values table record header type

\*/

typedef struct {

/\*\* Attribute DB permissions and length \*/

attdb\_head\_t head;

/\*\* Attribute DB entry handle \*/

attdb\_handle\_t handle;

/\*\* UUID of the attribute \*/

uuid\_t uuid;

/\*\* Attribute Value \*/

uint8\_t value[1];

} attdb\_record\_head\_t;

/\*\* @typedef attdb\_handle\_t

\*

\* @brief This is the attribute DB handle type

\*/

typedef uint16\_t attdb\_handle\_t;

Referenced by attdb.h file

value\_len % 513 = abs\_value\_len

Ex, 512 % 513 = 512 Variable length (as value\_len = abs\_value\_len)

Ex, 513 % 513 = 0 Fixed length (as value\_len != abs\_value\_len)

The use of the 2-bit split is to achieve bit compression

and have the head of an attribute in 4-bytes in total.

\*/

/\* Macro to decode absolute max len \*/

#define ATTDB\_MAX\_VALUE\_LEN(head) ((head.mlen1<<8) | head.mlen0)

/\* Macro to decode absolute actual/value len \*/

#define ATTDB\_ACTUAL\_VALUE\_LEN(head) ((head.vlen1<<8) | head.vlen0)

# SERVICES 0x3

/\*\*

### \* @struct dm\_service\_t

\* @brief Bluetooth GATT service

\*/

typedef struct

{

uuid\_t service\_uuid;

uint8\_t attr\_idx;

uint8\_t attr\_count;

} dm\_service\_t;

#### [Uuid\_t](#_Uuid_t)

# Attribute abstraction - ATTRS 0x4

/\*\*

### \* @struct dm\_attr\_t

\* @brief Container for an attribute abstraction

\*/

typedef struct

{

uuid\_t uuid; /\*\*< UUID of the Char.Value \*/

uint16\_t pipe\_map; /\*\*< Pipes associated with this Value \*/

uint8\_t type : 2; /\*\*< Is this Char. or Descriptor \*/

uint8\_t state : 4; /\*\*< Pipe states \*/

uint8\_t gpflags : 2; /\*\*< @todo Dont know!!! \*/

attdb\_handle\_t handle\_value; /\*\*< Handle corresponding to Value \*/

attdb\_handle\_t handle\_cccd; /\*\*< Handle corresponding to CCCD Attribute \*/

} dm\_attr\_t;

#### /\*\* @typedef attdb\_handle\_t

\*

\* @brief This is the attribute DB handle type

\*/

typedef uint16\_t attdb\_handle\_t;

# Vendor Specific UUIDS - VSUUIDS 0x5

### uint128\_le\_t

typedef unsigned char uint128\_le\_t[16];

# EXTATTRS 0x6

/\*\*

### \* @struct dm\_ext\_attr\_t

\* @brief Extended attribute info information

\*/

typedef struct

{

union

{

uint16\_t handle\_sccd; /\* local only \*/

dm\_cpf\_id\_t cpf\_id; /\* remote only \*/

} loc\_rem;

} dm\_ext\_attr\_t;

/\*\*

#### \* @struct dm\_cpf\_id\_t

\* @brief Bluetooth Characteristic Presentation Format ID

\*/

typedef struct

{

uint8\_t name\_space;

uint16\_t description;

} dm\_cpf\_id\_t;

# Custom advertising types - CUSTADTYPES 0x7

The target we want to modify is the custom advertising type.

The value of target “CUST\_AD\_TYPES” is 0x70

Look for the message <Len> 0x06 0x70 0x00 <Custom advertising data follows>

Interpret the bytes following the above using the structure below.

/\*\*

### \* @struct custom\_ad\_type\_t

\* @brief Custom ad type structure

\*/

typedef struct  
{  
uint8\_t type;  
uint8\_t len;  
uint8\_t adv\_data[ACI\_CUSTOM\_AD\_TYPE\_MAX\_DATA\_LENGTH];  
} dm\_custom\_ad\_type\_t;

#define ACI\_CUSTOM\_AD\_TYPE\_MAX\_DATA\_LENGTH 20

# CRC

The CRC is done on all setup ACI packets including the opcode and length excluding only the last 2 bytes which is the CRC.

uint16\_t crcproduct = 0x0000, data\_len = 0x0000;

uint16\_t crc\_16\_ccitt(uint16\_t crc, uint8\_t \* data\_in, uint16\_t data\_len) {

uint16\_t i;

for(i = 0; i < data\_len; i++)

{

crc = (unsigned char)(crc >> 8) | (crc << 8);

crc ^= data\_in[i];

crc ^= (unsigned char)(crc & 0xff) >> 4;

crc ^= (crc << 8) << 4;

crc ^= ((crc & 0xff) << 4) << 1;

}

return crc;

}

# Example ble\_modify\_setup\_data

The goal of the ble\_modify\_setup\_data example is to remove the Over The Air (OTA) write for the Device Name Characteristic. Structures for other targets will also be presented in this document.

Start with the OTA write enabled and from application controller . (In the nRFgo studio settings in nRF8001 configuration -> GAP settings -> Writeable Device name.)

To remove the OTA write alone, we need to change the Characteristic properties of the Device Name characteristic to only Read.

We also need to change the permissions of the Device Name Characteristic value to have only the Read Permission.

## CRC used:

Polynomial for CRC-16-CCITT https://upload.wikimedia.org/wikipedia/en/math/d/0/2/d0279fbd8e8844f8cbef81a76f006b93.png([X.25](https://en.wikipedia.org/wiki/X.25), [V.41](https://en.wikipedia.org/wiki/V.41), [HDLC](https://en.wikipedia.org/wiki/HDLC), [XMODEM](https://en.wikipedia.org/wiki/XMODEM), [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth), [PACTOR](https://en.wikipedia.org/wiki/PACTOR), [SD](https://en.wikipedia.org/wiki/Secure_Digital_card), many others; known as CRC-CCITT)

Initial seed value for the CRC is FFFF

The CRC is done on all setup ACI packets including the opcode and length excluding only the last 2 bytes which is the CRC.

## ATTDB 0x20

The format of all the Setup messages generated by the nRFgo studio is

<Len> <Opcode = Setup i.e. 0x06> <Target> <Offset within the Target>

The Target we want to modify is the "ATTDB".

The value of target "ATTDB" = 0x20.

The device name is usually the first message in this target so the offset we use is 0x00.

Look for the message <Len> 0x06 0x20 0x00.

{0x00,\

{\

{ header } { attdb\_head\_t }{handle}{UUID }{Value } **0x1f,0x06,0x20,0x00**,0x04,0x04,0x02,0x02,0x00,0x01,0x28,0x00,0x01,0x00,0x18,0x04,0x04,0x05,0x05,0x00,\

0x02,0x28,0x03,0x01,0x0e,0x03,0x00,0x00,0x2a,0x04,0x14,0x14,\

},\

},\

Change 0x14 to 0x04 for the permission of the value of the Device Name to be changed to Read.

After this is done; regenerate the CRC and update the CRC in the setup data. The last 2 bytes of the setup data is the CRC. The CRC is done on all setup ACI packets including the length and opcode excluding only the last 2 bytes which is the CRC. Ignore the 0x00 for all CRC calculations.

To change the Read permission bits of any 1 Characteristic value (with Read in its Characteristic property) from "plaintext" (which is created from the GATT Server marked as "No Security Required", to "authenticated encrypted" b11.

Modify the 2 Read Permissions bits for the Characteristic Value

uint8\_t rd:2;         /\*\*< Read  permisions (none, plaintext, authenticated plaintext, authenticated encrypted) \*/

You also need to set the Device setting bits which map to nRF8001 configuration -> Device Security back to “Security Required”

You can find the bits by also comparing the Setup between "Security Required" and "No Security Required".

The message uses the structure as below

/\*\* @file

\*

\* @brief This header file contains ATT DB internl data types

\* Inside the ATTDB, handles and UUIDs are stored in BE and values in LE

\* Please note that the ATT protocol requires data in LE format

\* ATTDB\_P2D macros store data in BE, for handles and UUIDs

\* uuid\_encode with UUID\_P2U\_UINT16 stores data in LE, for values

\*/

### attdb\_head\_t

/\*\* @brief Attribute permissions and length field

\*/

typedef struct {

/\* octet 0 \*/

uint8\_t mlen1:2; /\*\*< MSb of max len, refer Macros section in attdb.c \*/

uint8\_t valid:1; /\*\*< Valid record \*/

uint8\_t bcast:1; /\*\*< Broadcastable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t notify:1; /\*\*< Notifiable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t ind:1; /\*\*< Indicatable record, functionality implemented in upper layer, present for info purposes \*/

uint8\_t flighty:1; /\*\*< Flighty/volatile record \*/

uint8\_t locked:1; /\*\*< Lock (RFI) \*/

/\* octet 1 \*/

uint8\_t vlen1:2; /\*\*< MSb of actual len, refer Macros section in attdb.c \*/

uint8\_t rd:2; /\*\*< Read permisions (none, plaintext, authenticated plaintext, authenticated encrypted) \*/

uint8\_t wr:2; /\*\*< Write permisions (none, plaintext, authenticated plaintext, authenticated encrypted) \*/

uint8\_t ard:1; /\*\*< Read authorization (RFI) \*/

uint8\_t awr:1; /\*\*< Write authorization (RFI) \*/

/\* octet 2 \*/

uint8\_t mlen0:8; /\*\*< LSb of max len, refer Macros section in attdb.c \*/

/\* octet 3 \*/

uint8\_t vlen0:8; /\*\*< LSb of actual len, refer Macros section in attdb.c \*/

} attdb\_head\_t;

### /\*\* @struct attdb\_record\_head\_t

\*

\* @brief Attribute DB values table record header type

\*/

typedef struct {

/\*\* Attribute DB permissions and length \*/

attdb\_head\_t head;

/\*\* Attribute DB entry handle \*/

attdb\_handle\_t handle;

/\*\* UUID of the attribute \*/

uuid\_t uuid;

/\*\* Attribute Value \*/

uint8\_t value[1];

} attdb\_record\_head\_t;