TWN4

BLE Implementer's Guide

DocRev4, November 7, 2022
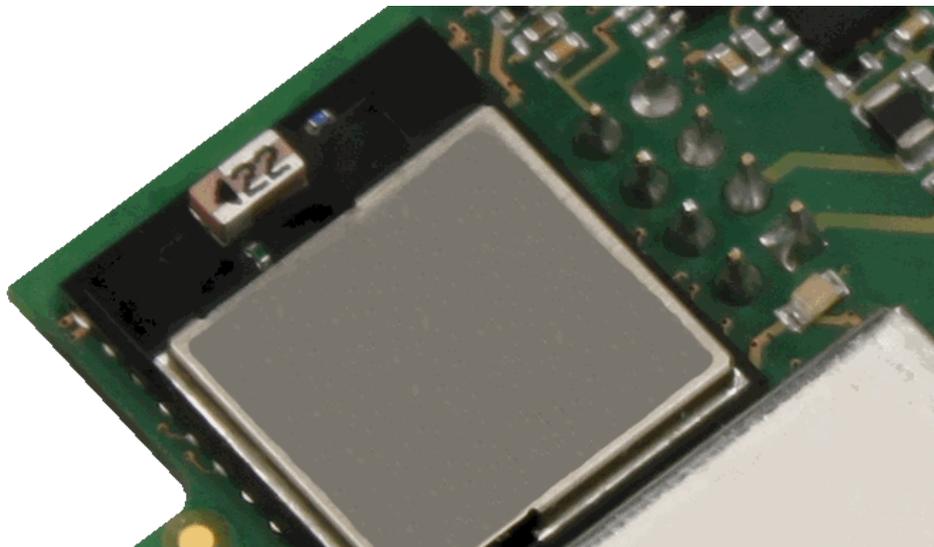
ELATEC GmbH

Contents

1 Introduction

In general, BLE integration into the TWN4 BLE reader family is performed via the API interface. The individual API functions control the BLE controller. The following graphic shows the TWN4 BLE Open System Interconnection model as an overview.

TWN4 BLE Open Systems Interconnection Model
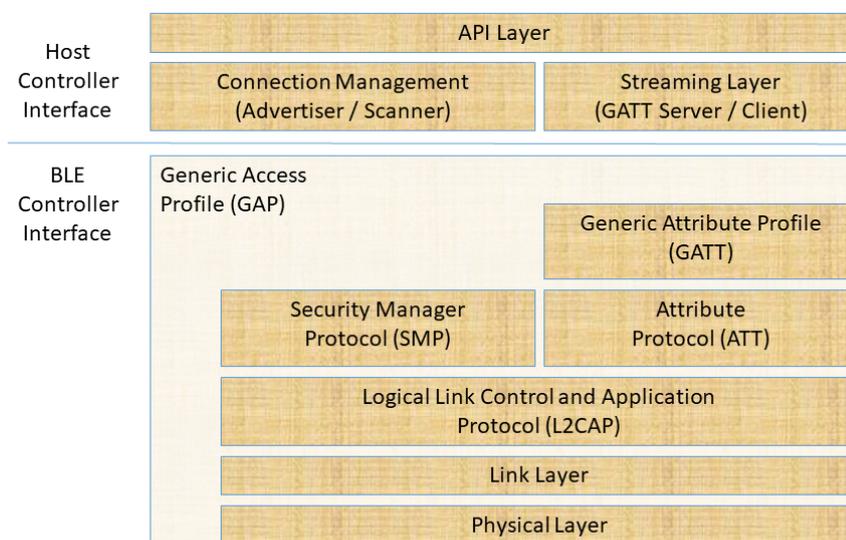
2 Mobile Badge BLE NFC

2.1 TWN4 Firmware

"Mobile Badge BLE NFC" is a virtually programmed card which runs as an application on a phone. The app connects to the TWN4 reader and presents the reader with a GATT server for reading and writing. The authentication process is an AES128 process with two ways to securely authenticate the device.

2.1.1 Created with AppBlaster

The AppBlaster program is a simple tool for configuring the reader to read and write various cards (LF, HF and BLE).

For the special Bluetooth card "Mobile Badge BLE NFC" configure the project to search for `Transponder type` "BLE (Bluetooth Low Energy)" -> "Mobile Badge BLE NFC".

Start the program AppBlaster, select "Configurable project" and configure all required transponders. For more detailed operation of AppBlaster, refer to the user manual of this software.
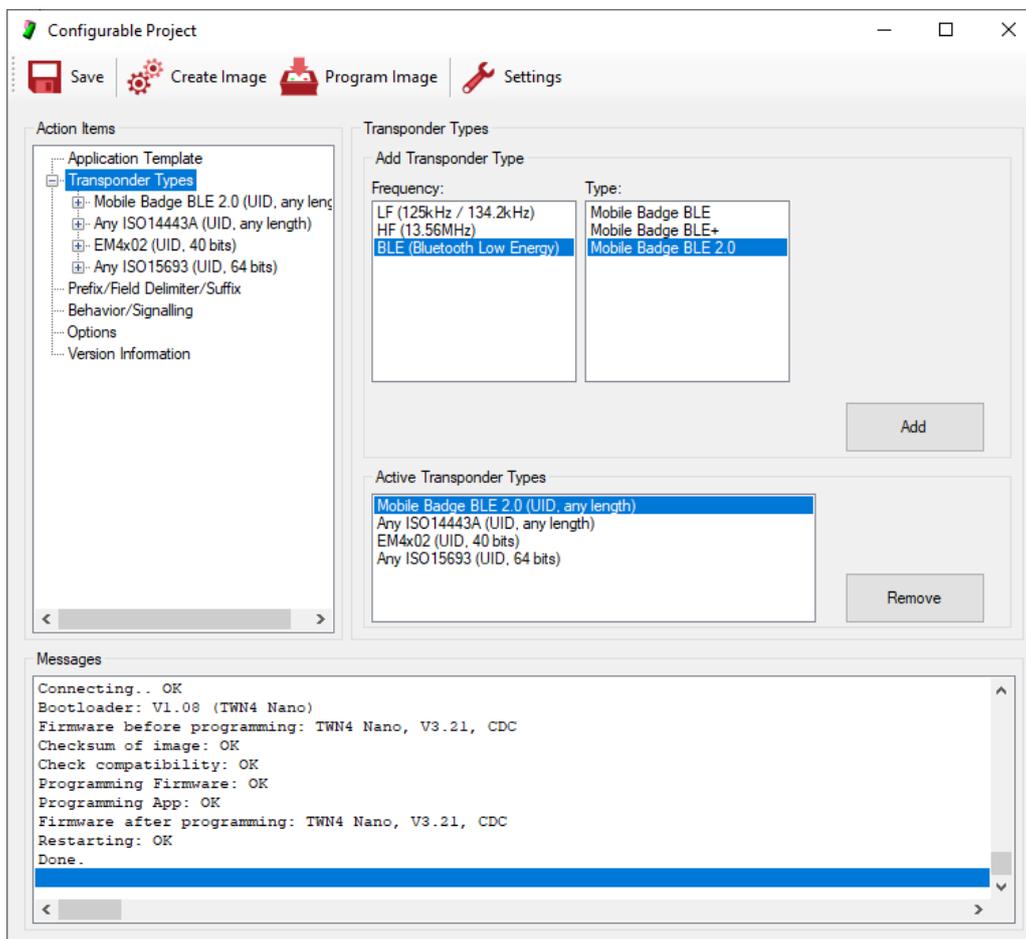
Next, select "Create Image" and "Program Image" to program the connected TWN4 reader. The reader now reads your selected transponder and the transponder "Mobile Badge BLE NFC" (e.g. the phone app on Android or iPhone).

2.1.2 Reader with Simple Protocol

The TWN4 Reader's Simple Protocol is a very flexible way to control the TWN4 Reader over a serial connection.

Setup your reader with Simple Protocol for reading cards:
Flash the simple protocol firmware "TWN4_NCx321_PRS104_Nano_CDC_Simple_Protocol.bix" (or newer versions) with the software AppBlaster to your TWN4 BLE reader (the reader needs to have B option).

Start the Director.exe application and press "Connect" to connect to your TWN4 reader. Select the section "Function Test" and then API -> "BLE", Function -> "BLEInit" and Mode -> "BLE_MODE_MOBILE_BADGE_2_0". Press "Send" to execute the function BLEInit(BLE_MODE_MOBILE_BADGE_2_0). Now the reader is configured to interact with "Mobile Badge BLE NFC" applications (for example, an app on the phone).
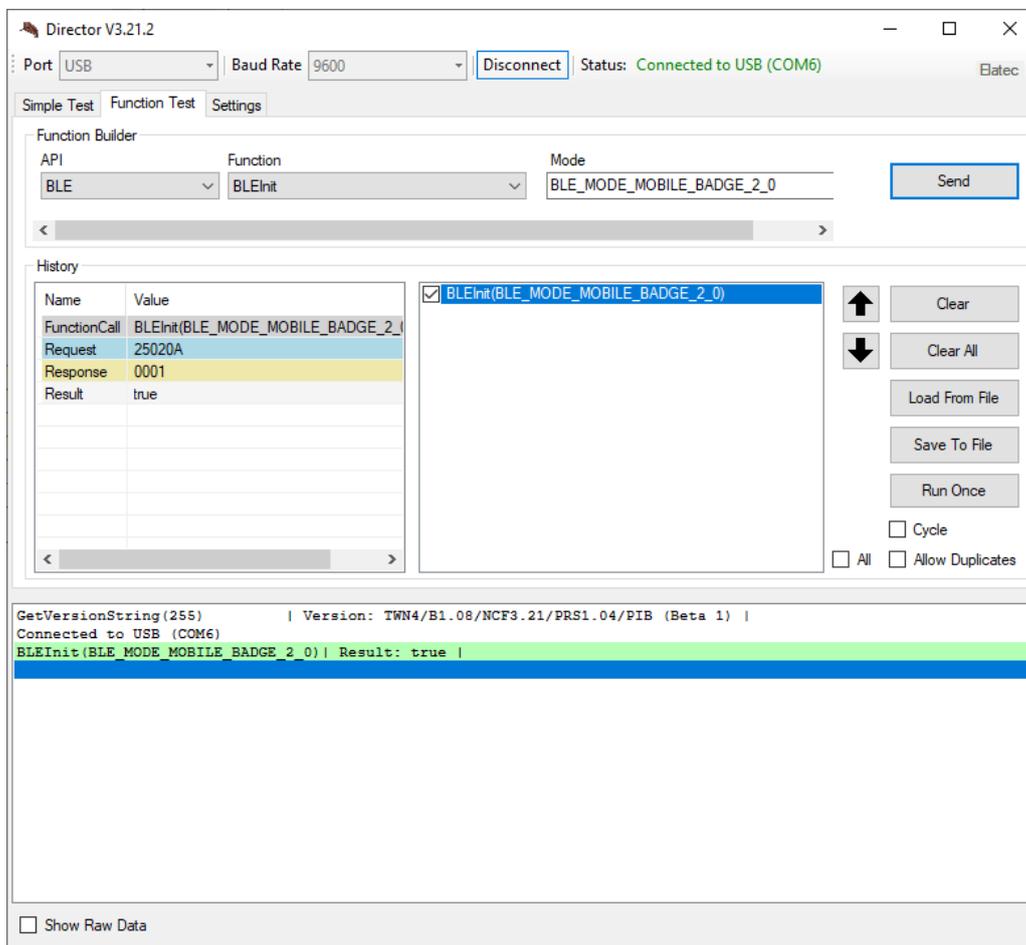
Go to the "Simple Test" section. Under "Select Tag Types", in addition to the other transponders, activate the type "BLE" in the list. Activate "Cycle" and the reader searches for the selected transponders. The screen shows you the IDs of your presented transponder.
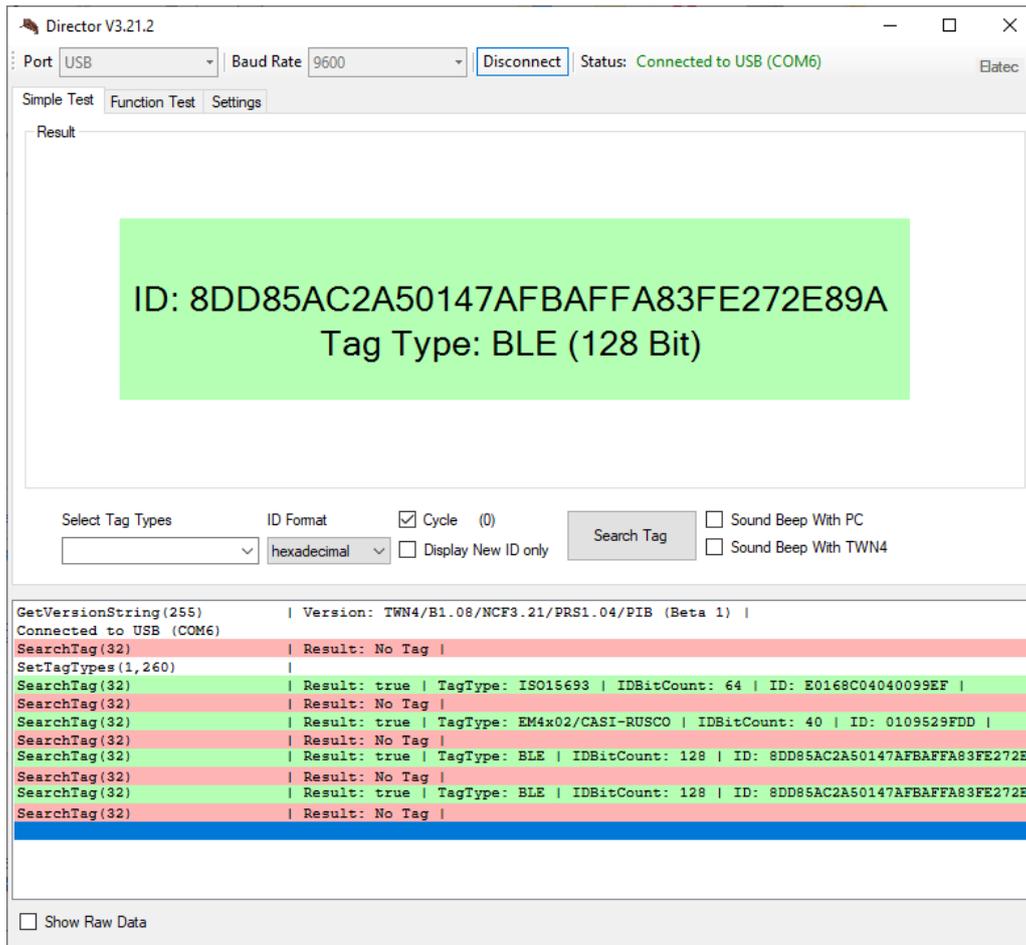
Note: The ID of Tag Type BLE is 128 bits (= 16 bytes).

The length of the data sent by the app depends on the application. The minimum data length is 8 bytes.

2.1.3 Third Party Reader App

The reading of cards is done with the Standard app. To read "Mobile Badge BLE NFC" cards, the app must be extended with the following source code:

```c
...

int main(void)
{
        OnStartup();
        ...
        SetTagTypes(LFTAGTYPES,HFTAGTYPES & ~BLE_MASK);
        ...
        if(BLEInit(BLE_MODE_MOBILE_BADGE_2_0))
        {
                OnBLEStarted();
        }

        // Main loop
        while (true)
        {
                ...
        }
}
```

2.2 Phone Applications

2.2.1 Android: Mobile Badge BLE NFC

Please download "Mobile Badge BLE NFC" from Play Store, install the software, start the app and accomplish your authentication.

2.2.2 iPhone: Mobile Badge BLE NFC

Please download "Mobile Badge BLE NFC" from Apple Store, install the software, start the app and accomplish your authentication.

2.2.3 Third Party Application

The authentication procedures with TWN4 BLE reader can be implemented in third party applications. The following steps helps with implementations.

2.2.3.1 Advertise-String

The TWN4 reader in the initialized mode "Mobile Badge BLE NFC" advertises with a preset reader string, which you can change with your own information.

The default advertiser string sent by the reader has the following properties: "Bluetooth Low Energy" device, reader name "ELATEC" and the manufacturer block with SIG ID 0x0752 (= "ELATEC GmbH").

To change this environment, use the system function `BLEPresetUserData`.

The advertise string is built up as follows:

Length	Data Type Value	Content	Description
2	0x01 (Flags)	0x06	
9	0xFF (Manufacturer Specific Data)	0x52 0x07	0x0752=SIG ID: ELATEC GmbH
		0x01	Block descriptor/Version
		0x02	Firmware version compatibility index
		0x01	Device type (1=BGM111; 2=BGM121; 3=BGM11S)
		0x01	Application type (1=short range; 2=long range)
		0x01	Application specific code (0=no encryption, 1=AES128)
		0x00	TX-Power 0=0.0dBm (BGM111), 80=8.0dBm (BGM121, BGM11S)
17	0x07 (Service Class UUID)	0xF0 0x34 0x9B 0x5F 0x80 0x00 0x00 0x80 0x00 0x10 0x00 0x00 0x1D 0xB8 0x00 0x00	UUID, 128 bit

TWN4 firmware version 3.21 and before is using a different advertiser string, which is documented here for reference and in order to maintain backward compatibility:

Length	Data Type Value	Content	Description
2	0x01 (Flags)	0x06	
7	0x09 (Complete Local Name)	0x45 0x4C 0x41 0x54 0x45 43	"ELATEC"
14	0xFF (Manufacturer Specific Data)	0x52 0x07	0x0752 = SIG ID: ELATEC GmbH
		0x01	Block descriptor/version
		0x01	Device type (1: TWN4 MultiTech 2; 2: TWN4 MultiTech 3; 3: TWN4 Palon)
		0x00 0x01	Application code (Mobile Badge = 0x0001; Cable Repl. = 0x0002)
		0x00 0x01	Transmission code
		0x00 0x01	Application specific code (0=no encryption; 1=AES128)
		0x00	TX-Power 0=0.0dB
		0x00 0x00	reserved

2.2.3.2 GATT Server

The TWN4 reader is the GATT client. To implement a data exchange, create your own GATT server with own GATT service and GATT characteristic in the application. Set the UUIDs for the GATT server and the GATT characteristic. The data length for the characteristic should be equal to or greater than 16 bytes.

To set the UUIDs for GATT service and characteristic in the TWN4 Reader software, use the system function `BLEStreamingUUID`.

2.2.3.3 AES128 MasterKey

When the connection between the GATT server and the GATT client is established, the authentication process begins. AES128 authentication requires a 16-byte master key.

To set the master key use the system function `BLESecuritySetOob` with parameter `BLE_SET_MOBILE_BADGE_MASTERKEY`.

2.2.3.4 The Authentication Process

Call system function `BLEInit` with the parameter `BLE_MODE_MOBILE_BADGE_2_0` and the fetch/read routine with the system function `SearchTag`.

Use the application to connect to the TWN4 reader (the default requirement is "Bluetooth Low Energy" Type, Reader Name "ELATEC", Manufacturer ID "ELATEC", and RSSI in the desired range).

Now the TWN4 reader checks the GATT service and GATT characteristic UUIDs and activates the notification on the GATT server characteristic.

The whole procedure with authentication sequence is shown in this diagram:
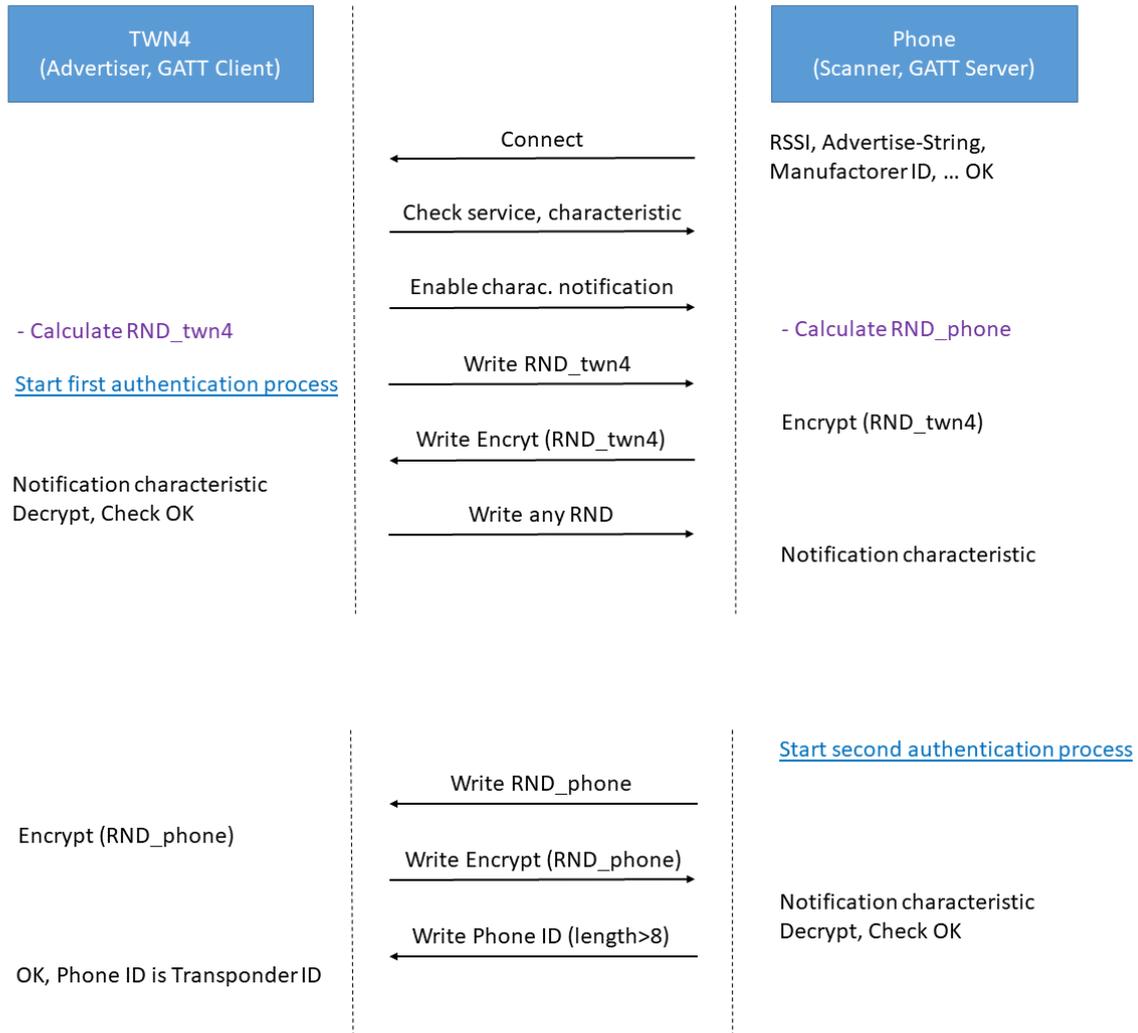
Annotation: RND_twn4 and RND_phone are 16 bytes random numbers.
The encryption and decryption is done using AES128 with the Master Key.
The phone ID sent from your application must be at least 8 bytes long.

3 Cable Replacement

The streaming mode is an operating mode for transferring individual bytes or blocks (up to 200 bytes at once) from one device to another and vice versa. One device is a TWN4 BLE reader and the other a second TWN4 device, a phone, a PC or another device.
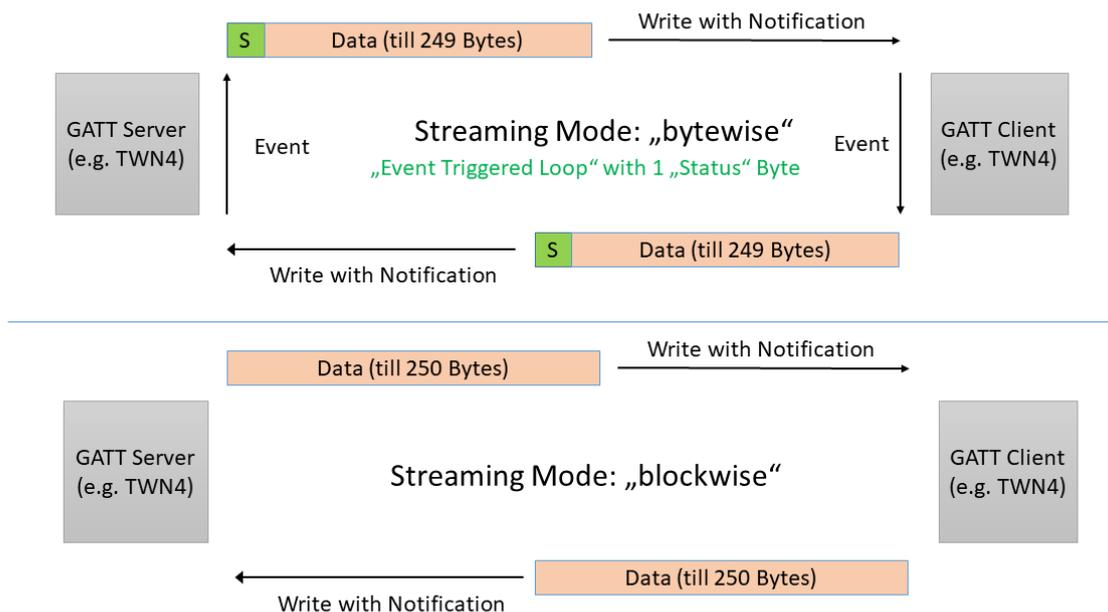
3.1 Datatransfer = BLOCKWISE

3.1.1 Simple Protocol Reader with Third Party Reader Apps

This chapter describes transferring Simple Protokol commands beetween two TWN4 BLE reader. One reader is connected to the PC and acts like a gateway which transfers commands between USB and BLE. The other TWN4 is executing the Simple Protocol commands sent by Director.

3.1.1.1 Reader A as Advertiser, GATT Server and Simple Command Executer

The reader A is a BLE Advertiser and has the GATT Server for data saving. Following parts of the source code describes this in outline:

```
void OnStartup(void)
{
        LEDInit(REDLED | YELLOWLED | GREENLED);
        LEDOff(REDLED);
```

```
        LEDOff(YELLOWLED);
        LEDOn(GREENLED);
        SetVolume(30);
}

// BLE Mode
const TBLEConfig BleConfig = {
        .ConnectTimeout = 0,
        .Power = 50,
        .BondableMode = 0xA1,
        .AdvInterval = 0x00a0,
        .ChannelMap = 0x07,
        .DiscoverMode = 0x04,
        .ConnectMode = 0x02,
        .SecurityFlags = 0x00,
        .IOCapabilities = 0x00,
        .Passkey = 0x00000000,
};

// User data
byte userdata[30] = {
        0x02,           // Length
        0x01,           // # Flags
        0x06,           // Device in LE General discoverable mode

        0x09,           // Length
        0x09,           // # Name
        'T','W','N','4','␣','B','L','E',

        0x0E,           // Length of the Manufacturer Data field
        0xff,           // Data type - manufacturer specific data - Manufacturer Data field
        0x52,0x07,      // Manufacturer data, Company ID field - 0x0752 = ELATEC GmbH SIG
        // Application Code 1 (example from ELATEC)
        0x00,0x02,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};

const byte UUID_SPP[16] = {0x8e,0xdf,0xae,0x3d,0x9b,0xcd,0x0e,0x88,0x74,0x42,0x12,0x41,0x04,
        0xc0,0x44,0x5a};
const byte UUID_SPP_DATA[16] = {0x52,0xc7,0x9e,0x16,0x9d,0x48,0x22,0xaa,0x43,0x4c,0x0a,0x2f,
        0xdf,0x9e,0xc2,0x43};

// Main program
int main(void)
{
        OnStartup();

        BLEPresetUserData(0, userdata, sizeof(userdata));
        BLEPresetConfig(&BleConfig);
        if(BLEInit(BLE_MODE_CUSTOM))
            BeepHigh();

        if(BLESetStreamingUUID(UUID_SPP, 16, UUID_SPP_DATA, 16))
                BeepHigh();

        if(BLESetStreamingMode(BLE_STREAM_CONN_ADVERTISE, BLE_STREAM_GATT_SERVER,
                    BLE_STREAM_TRANSFER_BLOCKWISE))
                BeepHigh();
```

```
int HostChannel = CHANNEL_BLE;
// Simple Protocol is running in ASCII mode W/O CRC.
SimpleProtoInit(HostChannel,PRS_COMM_MODE_ASCII | PRS_COMM_CRC_OFF);

// Main loop
while (true)
{
        if (SimpleProtoTestCommand())
        {
                // SimpleProtoMessage (MessageLength now contains command from host)
                SimpleProtoExecuteCommand();
                // SimpleProtoMessage (MessageLength now contains response to host)
                SimpleProtoSendResponse();
        }
}
}
```

3.1.1.2 Reader B as Scanner, GATT Client and Connected to Director.exe

The Reader B is a BLE scanner and a GATT client. The BLE scanner searches for the advertiser string from the Reader A and connects to it when the RSSI is in the near field. Discovering and checking UUIDs is not done in the example code. Once connected, the CHANNEL_USB of Reader B is the transmitting channel of the telegrams in both directions.

```
void OnStartup(void)
{
        LEDInit(REDLED | YELLOWLED | GREENLED);
        LEDOff(REDLED);
        LEDOff(YELLOWLED);
        LEDOn(GREENLED);
        SetVolume(30);
}

const byte UUID_SPP[16] = {0x8e,0xdf,0xae,0x3d,0x9b,0xcd,0x0e,0x88,0x74,0x42,0x12,0x41,0x04,
        0xc0,0x44,0x5a};
const byte UUID_SPP_DATA[16] = {0x52,0xc7,0x9e,0x16,0x9d,0x48,0x22,0xaa,0x43,0x4c,0x0a,0x2f,
        0xdf,0x9e,0xc2,0x43};

// Main program
int main(void)
{
        OnStartup();

        TBLEUUID BLEUUID;
        bool BLEStatus = 0;
        if(BLEInit(BLE_MODE_DISCOVER))
                BeepHigh();

        if(BLESetStreamingUUID(UUID_SPP, 16, UUID_SPP_DATA, 16))
                BeepHigh();

        if(BLESetStreamingMode(BLE_STREAM_CONN_DISCOVER, BLE_STREAM_GATT_CLIENT,
                        BLE_STREAM_TRANSFER_BLOCKWISE))
                BeepHigh();

        // Main loop
```

```
while (true)
{
	switch(BLECheckEvent())
	{
		case BLE_EVENT_LE_GAP_SCAN_RESPONSE:
		{
			BLEStatus=0;
			{
				byte DeviceRole,SecurityMode,RSSIRaw;
				BLEGetEnvironment(&DeviceRole,&SecurityMode,&RSSIRaw);
				byte rssi_tmp = 128 - (RSSIRaw&0x7F);
				if(rssi_tmp<50)
				{
					BLEDiscover(BLE_DISC_STOP_PHY_1M, 0, &BLEUUID);
					byte DeviceAddress[6], Address[6], AdressType;
					BLEGetAddress(DeviceAddress,Address,&AdressType);
					if(BLEConnectToDevice(Address, AdressType))
					{
						BeepHigh();
						BLEStatus = 1;
					}
					break;
				}
			}
			break;
		}
	}

	// TWN4 Connected :-)
	if(BLEStatus==1)
	{
		while(true)
		{
			while(!TestEmpty(CHANNEL_BLE, DIR_IN))
			{
				byte ch = ReadByte(CHANNEL_BLE);
				WriteByte(CHANNEL_USB, ch);
			}
			while(!TestEmpty(CHANNEL_USB, DIR_IN))
			{
				byte ch = ReadByte(CHANNEL_USB);
				WriteByte(CHANNEL_BLE, ch);
			}
			// connection lost?
			if(!BLECommand(BLE_CONN_STREAM_AVAILABLE,1))
			{
				BLEStatus=0;
				BeepLow();
				break;
			}
		}
	}
}
}
```

3.1.2 Transparent Data Transfer with 2 Readers

This chapter describes how to transmit data in a transparent mode between two TWN4 BLE readers. Both readers are connected with e.g. HTerm (terminal program) for individual data transmission. The sample code shows the configuration in **block** mode.

3.1.2.1 Reader A as Advertiser, GATT Server with Transparent Data Transfer

Reader A is a BLE Advertiser and has the GATT server for data transmission. The source code is similar to the example of the Simple Protocol. The different parts are shown below.

```
// Main program
int main(void)
{
        OnStartup();

        BLEPresetUserData(0, userdata, sizeof(userdata));
        BLEPresetConfig(&BleConfig);
        if(BLEInit(BLE_MODE_CUSTOM))
                OnBLEStartup();

        if(BLESetStreamingUUID(UUID_SPP, 16, UUID_SPP_DATA, 16))
                BeepHigh();

        if(BLESetStreamingMode(BLE_STREAM_CONN_ADVERTISE, BLE_STREAM_GATT_SERVER,
                        BLE_STREAM_TRANSFER_BLOCKWISE))
                BeepHigh();

        // Main loop
        while (true)
        {
                if(!TestEmpty(CHANNEL_BLE, DIR_IN))
                {
                        byte ch = ReadByte(CHANNEL_BLE);
                        WriteByte(CHANNEL_USB, ch);
                }
                if(!TestEmpty(CHANNEL_USB, DIR_IN))
                {
                        byte ch = ReadByte(CHANNEL_USB);
                        WriteByte(CHANNEL_BLE, ch);
                }
        }
}
```

3.1.2.2 Reader B as Scanner, GATT Client with Transparent Data Transfer

The Reader B is a BLE scanner and a GATT client. The BLE scanner searches for the advertiser string from Reader A and connects to it when the RSSI is in the near field. Once connected, the CHANNEL_USB of the Reader A and B is the transmitting channel of telegrams in both directions.

The source code is the same as the Scanner/GATT Client example of the Simple Protocol.

3.2 Datatransfer = BYTEWISE

3.2.1 Transparent Data Transfer with 2 Readers

This chapter describes how to transmit data in a transparent mode between two TWN4 BLE readers. Both readers are connected with e.g. HTerm (terminal program) for individual data transmission. The sample code shows the configuration in **byte** mode.

3.2.1.1 Reader A as Advertiser, GATT Server with Transparent Data Transfer

The Reader A is a BLE Advertiser with the GATT server for data transmission. The source code is like the example of the Simple Protocol. The different part is shown below.

```
...
if(BLESetStreamingMode(BLE_STREAM_CONN_ADVERTISE, BLE_STREAM_GATT_SERVER,
            BLE_STREAM_TRANSFER_BYTEWISE))
        BeepHigh();
...
```

3.2.1.2 Reader B as Scanner, GATT Client with Transparent Data Transfer

The Reader B is a BLE scanner and a GATT client. The BLE scanner searches for the advertiser string from the Reader A and connects to it when the RSSI is in the near field. Once connected, the CHANNEL_USB of the Reader A and B is the transmitting channel of telegrams in both directions.

The source code is similar to the Scanner/GATT Client example of the Simple Protocol. The different part is shown below.

```
...
if(BLESetStreamingMode(BLE_STREAM_CONN_ADVERTISE, BLE_STREAM_GATT_SERVER,
            BLE_STREAM_TRANSFER_BYTEWISE))
        BeepHigh();
...
```

3.2.2 Data Transfer with Two Readers programmed with Simple Protocol

This chapter describes a data transfer with 2 TWN4 BLE readers, which are operated with Simple Protocol.

3.2.2.1 Reader A as Advertiser, GATT Server

The Reader A is a BLE Advertiser with the GATT Server for data transfer. Test the configuration with Director.exe. The following screenshot shows an example of initializing, reading and writing bytes.
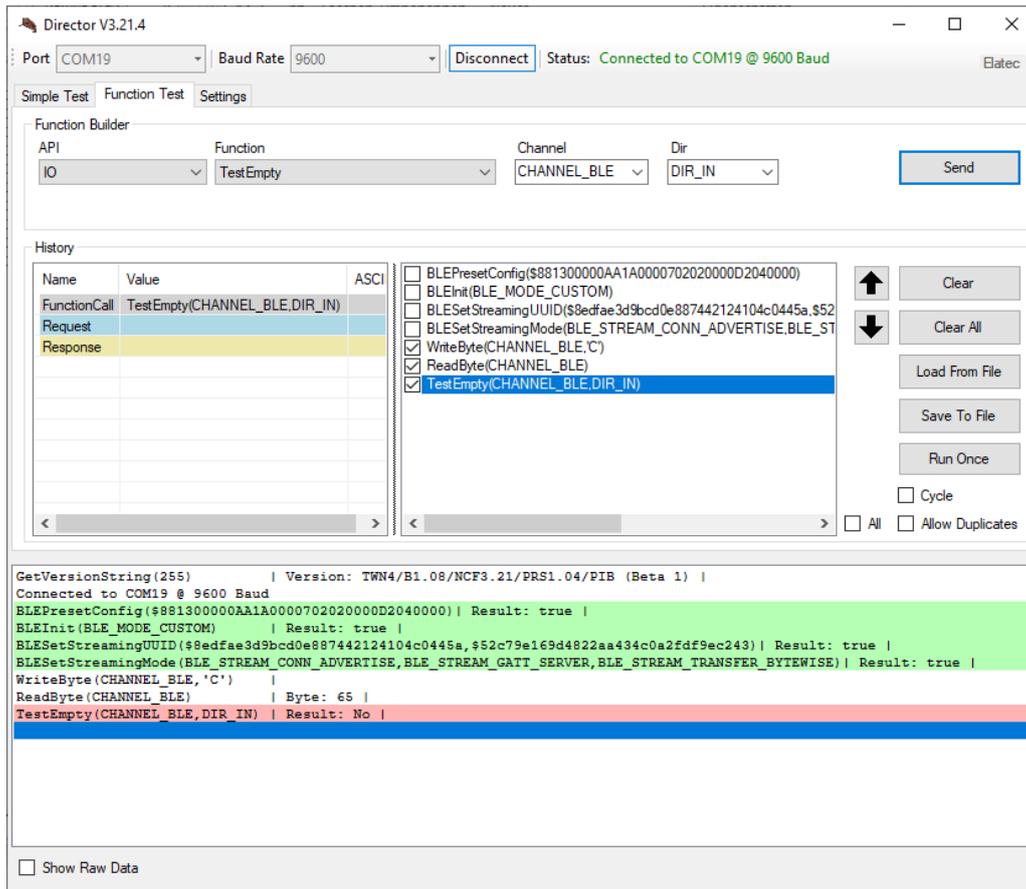
Once:

```
BLEPresetConfig($881300000AA1A0000702020000D2040000)
BLEInit(BLE_MODE_CUSTOM)
BLESetStreamingUUID($8edfae3d9bcd0e887442124104c0445a,$52c79e169d4822aa434c0a2fdf9ec243)
BLESetStreamingMode(BLE_STREAM_CONN_ADVERTISE,BLE_STREAM_GATT_SERVER,
        BLE_STREAM_TRANSFER_BYTEWISE)
```

Cycle:

```
TestEmpty(CHANNEL_BLE, DIR_IN)
ReadByte(CHANNEL_BLE);
WriteByte(CHANNEL_BLE, ch);
```

Note that function ReadByte(..) is a blocking function!

3.2.2.2 Reader B as Scanner, GATT Client

The Reader B is a BLE scanner and a GATT client for data transfer. Testing the configuration with Director.exe is possible. After a connection is established, the read and write bytes are identical to the example of the advertiser.

Once:

```
BLEInit(BLE_MODE_DISCOVER)
BLESetStreamingUUID($8edfae3d9bcd0e887442124104c0445a,$52c79e169d4822aa434c0a2fdf9ec243)
```

```
Cycle for scan Advertiser reader and connect:

BLECheckEvent()
BLEGetEnvironment()      -> for RSSI

Once for connection:

BLEDiscover(BLE_DISC_STOP_PHY_1M, 0,$100000000000000080000000000000000)
BLEGetAddress()
BLEConnectToDevice()
BLESetStreamingMode(BLE_STREAM_CONN_DISCOVER,BLE_STREAM_GATT_CLIENT,
        BLE_STREAM_TRANSFER_BYTEWISE)

Cycle:

TestEmpty(CHANNEL_BLE, DIR_IN)
ReadByte(CHANNEL_BLE);
WriteByte(CHANNEL_BLE, ch);
```

4 Disclaimer

ELATEC reserves the right to change any information or data in this document without prior notice. The distribution and the update of this document is not controlled. ELATEC declines all responsibility for the use of product with any other specifications but the ones mentioned above. Any additional requirement for a specific custom application has to be validated by the customer himself at his own responsibility. Where application information is given, it is only advisory and does not form part of the specification.

All referenced brands, product names, service names and trademarks mentioned in this document are the property of their respective owners.