DWM1001 Bluetooth API

USING BLE API FUNCTIONS TO CONFIGURE DWM1001 MODULE

This document is subject to change without notice

DOCUMENT INFORMATION

Disclaimer

Decawave reserves the right to change product specifications without notice. As far as possible changes to functionality and specifications will be issued in product specific errata sheets or in new versions of this document. Customers are advised to check the Decawave website for the most recent updates on this product

LIFE SUPPORT POLICY

Decawave products are not authorized for use in safety-critical applications (such as life support) where a failure of the Decawave product would reasonably be expected to cause severe personal injury or death. Decawave customers using or selling Decawave products in such a manner do so entirely at their own risk and agree to fully indemnify Decawave and its representatives against any damages arising out of the use of Decawave products in such safety-critical applications.

Caution! ESD sensitive device.
Precaution should be used when handling the device in order to prevent permanent damage

DISCLAIMER

(1) This Disclaimer applies to the software provided by Decawave Ltd. ("Decawave") in support of its DWM1001 module product ("Module") all as set out at clause 3 herein ("Decawave Software").

(2) Decawave Software is provided in two ways as follows: -
- (a) pre-loaded onto the Module at time of manufacture by Decawave ("Firmware");
- (b) supplied separately by Decawave ("Software Bundle").

(3) Decawave Software consists of the following components (a) to (d) inclusive:

- (a) The **Decawave Positioning and Networking Stack** ("PANS"), available as a library accompanied by source code that allows a level of user customisation. The PANS software is pre-installed and runs on the Module as supplied, and enables mobile "tags", fixed "anchors" and "gateways" that together deliver the DWM1001 Two-Way-Ranging Real Time Location System ("DRTLS") Network.

- (b) The **Decawave DRTLS Manager** which is an Android™ application for configuration of DRTLS nodes (nodes based on the Module) over Bluetooth™.

- (c) The **Decawave DRTLS Gateway Application** which supplies a gateway function (on a Raspberry Pi ®) routing DRTLS location and sensor data traffic onto an IP based network (e.g. LAN), and consists of the following components:
  - DRTLS Gateway Linux Kernel Module
  - DRTLS Gateway Daemon
  - DRTLS Gateway MQTT Broker
  - DRTLS Gateway Web Manager

- (d) **Example Host API functions**, also designed to run on a Raspberry Pi, which show how to drive the Module from an external host microprocessor.

(4) The following third party components are used by Decawave Software and are incorporated in the Firmware or included in the Software Bundle as the case may be: -

- (a) The PANS software incorporates the Nordic SoftDevice S132-SD-v3 version 3.0.0 (production) which is included in the Firmware and is also included in the Software Bundle;

- (b) The PANS software uses the eCos RTOS which is included in the Software Bundle. The eCos RTOS is provided under the terms of an open source licence which may be found at: http://ecos.sourceware.org/license-overview.html;

- (c) The PANS software uses an open source CRC-32 function from FreeBSD which is included in the Software Bundle. This CRC-32 function is provided under the terms of the BSD licence which may be found at: https://github.com/freebsd/freebsd/blob/386ddae58459341ec567604707805814a2128a57/COPYRIGHT;

- (d) The Decawave DRTLS Manager application uses open source software which is provided as source code in the Software Bundle. This open source software is provided under the terms of the Apache Licence v2.0 which may be found at http://www.apache.org/licenses/LICENSE-2.0;

(e) The Decawave DRTLS Gateway Application uses the following third party components: -

(i) The Linux Kernel which is provided as source code in the Software Bundle. The Linux Kernel is provided under the terms of the GPLv2 licence which may be found at: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html and as such the DWM1001 driver component of the DRTLS Gateway Application is provided under the same license terms;

(ii) The three.js JavaScript library, the downloadable version of which is available here https://threejs.org/, is provided under the terms of the MIT Licence which may be found at https://opensource.org/licenses/MIT.

Items (a), (b), (c), (d) and (e) in this section 4 are collectively referred to as the "Third Party Software"

(5) Decawave Software incorporates source code licensed to Decawave by Leaps s.r.o., a supplier to Decawave, which is included in the Firmware and the Software Bundle in binary and/or source code forms as the case may be, under the terms of a license agreement entered into between Decawave and Leaps s.r.o.

(6) Decawave hereby grants you a free, non-exclusive, non-transferable, worldwide license without the right to sub-license to design, make, have made, market, sell, have sold or otherwise dispose of products incorporating Decawave Software, to modify Decawave Software or incorporate Decawave Software in other software and to design, make, have made, market, sell, have sold or otherwise dispose of products incorporating such modified or incorporated software PROVIDED ALWAYS that the use by you of Third Party Software as supplied by Decawave is subject to the terms and conditions of the respective license agreements as set out at clause 4 herein AND PROVIDED ALWAYS that Decawave Software is used only in systems and products based on Decawave semiconductor products. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER DECAWAVE INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Decawave semiconductor products or Decawave Software are used.

(7) Downloading, accepting delivery of or using Decawave Software indicates your agreement to the terms of (i) the license grated at clause 6 herein, (ii) the terms of this Disclaimer and (iii) the terms attaching to the Third Party Software. If you do not agree with all of these terms do not download, accept delivery of or use Decawave Software.

(8) Decawave Software is solely intended to assist you in developing systems that incorporate Decawave semiconductor products. You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your systems and products. THE DECISION TO USE DECAWAVE SOFTWARE IN WHOLE OR IN PART IN YOUR SYSTEMS AND PRODUCTS RESTS ENTIRELY WITH YOU AND DECAWAVE ACCEPTS NO LIABILTY WHATSOEVER FOR SUCH DECISION.

(9) DECAWAVE SOFTWARE IS PROVIDED "AS IS". DECAWAVE MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO DECAWAVE SOFTWARE OR USE OF DECAWAVE SOFTWARE, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. DECAWAVE DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF ANY THIRD

PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO DECAWAVE SOFTWARE OR THE USE THEREOF.

(10) DECAWAVE SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON DECAWAVE SOFTWARE OR THE USE OF DECAWAVE SOFTWARE. IN NO EVENT SHALL DECAWAVE BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, INCLUDING WITHOUT LIMITATION TO THE GENERALITY OF THE FOREGOING, LOSS OF ANTICIPATED PROFITS, GOODWILL, REPUTATION, BUSINESS RECEIPTS OR CONTRACTS, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION), LOSSES OR EXPENSES RESULTING FROM THIRD PARTY CLAIMS. THESE LIMITATIONS WILL APPLY REGARDLESS OF THE FORM OF ACTION, WHETHER UNDER STATUTE, IN CONTRACT OR TORT INCLUDING NEGLIGENCE OR ANY OTHER FORM OF ACTION AND WHETHER OR NOT DECAWAVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF DECAWAVE SOFTWARE OR THE USE OF DECAWAVE SOFTWARE.

(11) You acknowledge and agree that you are solely responsible for compliance with all legal, regulatory and safety-related requirements concerning your products, and any use of Decawave Software in your applications, notwithstanding any applications-related information or support that may be provided by Decawave.

(12) Decawave reserves the right to make corrections, enhancements, improvements and other changes to its software, including Decawave Software, at any time.

Mailing address: Decawave Ltd.,
Adelaide Chambers,
Peter Street,
Dublin D08 T6YA
IRELAND.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Overview

This document describes the DWM1001 APIs available over the Bluetooth Low Energy (BLE) link.

In the DWM1001 BLE API design, the DWM1001 module is acting as the BLE peripheral which can communicate with BLE central through these APIs. This document introduces the APIs that the BLE central can use for the communication. An Android application, the Decawave RTLS Manager, is provided to exercise the BLE APIs.

1.2 Used terminology

Anchor: The DWM1001 module working as a node that has a fixed location.

Tag: The DWM1001 module working as a mobile node, determines dynamically its position with the help of anchors.

DWM1001 Network (or DRTLS or simply Network): a set of anchors and tags cooperating together. These anchors and tags are called *network nodes*. Each DWM1001 network has a unique 2-byte ID (PAN ID).

Network Nodes (or simply Nodes): anchors and tags (see DWM1001 Network).

1.3 Position Representation

In presenting locations and distances in a Real-Time Positioning System, there are two items to consider:

- Accuracy
- Precision

Accuracy is the error between the position reported by the nodes and the real position. DWM1001 performance is < 10 cm LOS (as quoted in the DWM1001 Datasheet)

Precision is the value a least-significant bit (LSB) represents. In the on-board firmware of this system (PANS), the precision is 1 mm, i.e. 0.001 meter. The positions are presented in 3-dimension coordinates (X, Y, Z), where each is a 32-bit integer. Each LSB represents 1 mm. This is for easier interpretation of the value as well as easier mathematics on the reported values.

When deciding on the precision parameter, it is important to choose it with respect to accuracy in order to get a meaningful result. It does not make sense to show the user precise values if the accuracy is low. A 1 mm precision is too fine-grained with respect to the current 10 cm accuracy. Therefore in the Android application, precision of 1 cm is used. Only when the coordinate/distance has changed over 1 cm will the updated value be sent to the Android application.

2 NETWORK NODE

The BLE central device connects directly with the network nodes to set up and retrieve parameters. It needs to connect to each device individually to configure/control it.

2.1 BLE GATT Model

The ***network node service*** UUID is **680c21d9-c946-4c1f-9c11-baa1c21329e7**. All characteristic values are encoded as little endian as the BLE specification suggests.

RW – read/write; RO – read only; WO – write only

2.1.1 Network Node Characteristics

uuid	name	length	value	flags
Std. GAP service, label **0x2A00**	Label	Var	UTF-8 encoded string	RW
3f0afd88-7770-46b0-b5e7-9fc099598964	Operation mode	2 bytes	See Section 2.1.2 for details on data encoding	RW
80f9d8bc-3bff-45bb-a181-2d6a37991208	Network ID	2 bytes	Unique identification of the network (PAN ID)	RW
a02b947e-df97-4516-996a-1882521e0ead	Location data mode	1 byte	0 - Position 1 - Distances 2 - Position + distances	RW
003bbdf2-c634-4b3d-ab56-7ec889b89a37	Location data	106 bytes max	See Section 2.1.3 for details on data encoding	RO
f4a67d7d-379d-4183-9c03-4b6ea5103291	Proxy position	76 bytes	Used by the module as a notification about new tag positions for the BLE central. See section 2.1.4 for more details.	RO
1e63b1eb-d4ed-444e-af54-c1e965192501	Device info	29 bytes	Node ID (8 bytes), HW version (4 bytes), FW1 version (4 bytes), FW2 version (4 bytes), FW1 checksum (4 bytes), FW2 checksum (4 bytes), RDonly Operation flags (1 byte)	RO
0eb2bc59-baf1-4c1c-8535-8a0204c69de5	Statistics	120 bytes	Node statistics	RO

5955aa10-e085-4030-8aa6-bdfac89ac32b	FW update push	Max 37 bytes	Used to send structured data (FW update packets) to the module (BLE peripheral), the size is set according to max transmission unit (MTU). See Section 2.3.4 for details on data encoding.	WO
9eed0e27-09c0-4d1c-bd92-7c441daba850	FW update poll	9 bytes	Used by the module as a response/notification for the BLE central, See Section 2.3.4 for details on data encoding	RO
ed83b848-da03-4a0a-a2dc-8b401080e473	Disconnect	1 byte	Used to explicitly disconnect from BLE peripheral by writing value=1 (workaround due to android behavior)	WO

Note: The label characteristic is a special one. It is part of the standard mandatory GAP service (0x1800) under the standard name characteristic (0x2A00).

2.1.2 Operation mode characteristic

Operation mode characteristic is of 2 bytes and contains the configuration information of the nodes. The format is defined as follows:

1st byte (bit 7 down to 0)	
Bit	**value**
7	tag (0), anchor (1)
6 - 5	UWB - off (0), passive (1), active (2)
4	firmware 1 (0), firmware 2 (1)
3	accelerometer enable (0, 1)
2	LED indication enabled (0, 1)
1	firmware update enable (0, 1)
0	reserved
2nd byte (bit 7 down to 0)	
Bit	**value**
7	initiator enable, anchor specific (0, 1)
6	low power mode enable, tag specific (0, 1)
5	location engine enable, tag specific (0, 1)
4 - 0	reserved

2.1.3 Location data characteristic

Location data characteristic can contain position, distances or both. The format of the position and distances are defined as follows:

type (1 byte)	value
0 - Position only	X,Y,Z coordinates (each 4 bytes) and quality factor (1 byte), total size: 13 bytes
1 - Distances	First byte is distance count (1 byte) Sequence of node ID (2 bytes), distance (4 bytes) and quality factor (1 byte) Max value contains 15 elements, size: 8 - 106
2 - Position and Distances	Encoded Position (as above, 13 bytes) Encoded Distances (as above, 8 - 29 bytes). Position and distances are sent by tag, with a maximum number of 4 ranging anchors

Note 1: the characteristic value might be completely empty (zero length) meaning that there are neither known positions nor known distances.

*Note 2: although **location data mode** includes position and distances, it is still possible to receive distances only in the characteristic in case when position is not known.*

2.1.4 Proxy positions characteristics

Proxy positions characteristics is provided to overcome limitation of concurrently connected nodes to the BLE central (mobile device). A passive node uses this characteristic to stream/notify about tag position updates.

Data are encoded in this characteristic as follows:

- *1 byte: number of elements (max 5)*
- *[sequence] tag position: 2 bytes node id, 13 bytes position*

Thus the maximum size of 5 tag positions is 76 bytes long.

2.1.5 Anchor-specific Characteristics

An anchor may operate as either an anchor or anchor initiator.

uuid	name	length	value	flags
3f0afd88-7770-46b0-b5e7-9fc099598964	Operation Mode	2 bytes	Bit 7 in 2nd byte: initiator enable (0, 1) (see Section 2.1.3 for detail)	RW
1e63b1eb-d4ed-444e-af54-c1e965192501	Device info		RD only operation flags: BXXXXXXX B: bridge 1/0	RO
f0f26c9b-2c8c-49ac-ab60-fe03def1b40c	Persisted position	13 bytes	X,Y,Z coordinates each 4-byte precision + quality factor (1 byte, value 1 - 100)	WO

28d01d60-89de-4bfa-b6e9-651ba596232c	MAC stats	4 bytes	Reserved for internal debug MAC statistics	RO
17b1613e-98f2-4436-bcde-23af17a10c72	Cluster info	5 bytes	Seat number (1 byte)/Cluster map (2 bytes)/Cluster neighbor map (2 bytes)	RO
5b10c428-af2f-486f-aee1-9dbd79b6bccb	Anchor list	33 bytes	list of node IDs, count (1 byte), sequence of 2 bytes node IDs, max 16 elements in list	RO

2.1.6 Tag-specific Characteristics

Each tag determines its own position based on the information sent by 4 surrounding anchors. The tag provides complete information of how its position is computed

uuid	name	length	value	flags
3f0afd88-7770-46b0-b5e7-9fc099598964	Operation Mode	2 bytes	Bit 6 in 2nd byte: low power mode enable (0, 1) Bit 5 in 2nd byte: location engine enable (0, 1) (see Section 2.1.3 for detail)	RW
7bd47f30-5602-4389-b069-8305731308b6	Update rate	8 bytes	Broadcast new position each *U1* ms when moving, broadcast new position each *U2* ms when stationary. U1 (4 bytes), U2 (4 bytes)	RW

2.2 BLE Advertisements

BLE advertisements are a common way for a peripheral device to let others know its presence. The broadcast payload is made of triplets according to BLE spec, i.e. [length, type, <data>]. Both anchors and tags will broadcast basic information about their **presence and operation mode**. The BLE advertisement is not long enough to also include the position info.

In BLE advertisement a maximum payload of 31 bytes can be used:

- First 3 bytes are mandatory flags (one AD triplet).
- The rest 28 bytes can be used by the app to fill in AD records (each record has 2 bytes overhead of length and type)

2.2.1 Presence Broadcast

The BLE on the DWM1001 module works in the connectable undirected mode. It advertises its presence with a presence broadcast which contains the availability of service and some service data.

The presence broadcast follows the BLE advertisement frame structure and makes use of the 28 bytes to present information.

Because the presence broadcast has connectable flag set to true, a shortened local name AD record of 8 bytes must be included to overcome potential Android BLE stack bug. (As described in [1]). The remaining bytes are filled with service data: 2 bytes for the AD record header, 16 bytes UUID, 1 byte shortened operation mode and 1 byte change counter.

The presence broadcast frame has 3 + 20 + 8 bytes in total, i.e. 31 bytes. The frame structure is shown in the table below.

AD triplet - part identification	value
LEN	0x02
TYPE	0x01 (Flags)
DATA	Device/Advertisement flags - connectable
LEN	0x13 (19 in decimal)
TYPE	0x21 (SERVICE_DATA)
DATA	**680c21d9-c946-4c1f-9c11-baa1c21329e7** (16 bytes)
	Bit layout: OXXEFFUU (1 byte) O - operation mode (tag 0, anchor 1) XX - reserved E - error indication FF - flags: initiator, bridge UU - UWB: off (0), passive (1), active (2)
	Change counter (1 byte) - change counter changes each time a characteristic gets changed (except for node statistics and specifically for Tag: position and ranging anchor)
LEN	0x07 (max)
TYPE	0x08 (Shortened local name)
DATA	First 6 letters (or less) of device local name as defined by GATT spec.

2.3 Firmware Update

The firmware update functionality is used to update the module's firmware. It can be performed either over UWB or over BLE. This section describes the control and data flow over BLE.

During FW update two characteristics, **FW update push** and **FW update poll**, are used to implement the request/response protocol.

2.3.1 Initiating FW Update

Steps:

- The *Android device* (BLE central) sets up an indication on **FW update poll** Client Characteristic Configuration Descriptor (CCCD).
- The *Android device* asks the network node if it is willing to perform the update by sending the update request/offer packet to **FW update push** characteristic. This initialization packet

contains: firmware version, firmware checksum, overall firmware binary size (in bytes). This is reliable write, aka. write with response.

- The *Network node* responds with indication on *FW update poll* in two cases:
  Case 1: YES, “send me the first data buffer”, see *Transmitting the FW binary* section for more information;

  Case 2: NO, and *error code* provides refuse reason.

2.3.1.1 Error states:

- **Android device**: received explicit NO indication along with error code/reason

Resolution: the *Android device* disables CCCD indication on *FW update poll* and notifies the upper layer about the refuse reason.

- **Network node**: sudden disconnect

Resolution: leave the FW update mode, reset current state as if the FW update did not happen.

- **Mobile device**: detects that connection has been closed.

Resolution: Retry. If still unsuccessful after 30 seconds from FW update initialization, report to upper layer. Let the user re-initiate the firmware update on request.

2.3.2 Transmitting the FW binary

This section is inspired by [2].

The data transmission is initiated by a network node. The network node tells the mobile device precisely which data buffer it wants using so called *FW buffer request*: size and offset. The mobile device starts sending the requested buffer in small chunks using write without response so there is no full round trip involved. The elementary chunk size is equal to MTU so that it fits into a single transmitted packet. The chunk consists of:

- Data: size should be rounded to power of 2. The current chunk size is set to 32 bytes.
- Relative offset (from the very beginning): 4 bytes.
- Identification of the message type: FIRMWARE_DATA_CHUNK (= 0x1): 1 byte

The data transmission is completely driven by the network node. After the data buffer is sent, the mobile device waits for further instructions. During the transmission, the network node normally asks for data buffers sequentially one by one to get a continuous byte sequence of firmware. The node might ask for an unexpected buffer if exceptions happen, for example the current buffer transmission fails.

2.3.2.1 Error states:

- **Network node**: data chunk is missing upon reception (non-continuous sequence), or out-of-order chunks.

Resolution: send *FW buffer request* specifying the missing chunk and the rest of the buffer.

- **Mobile device**: receives *FW buffer request* during ongoing data transmission.

Resolution: stop sending data, set current offset to the one in the *FW buffer request* and restart data transmission.

2.3.3 Finishing the transmission

Once the last data buffer has been successfully received, the network node will let the mobile device know via indication on *FW update poll* that it has received the full firmware binary. Upon its reception, the mobile device:

1. disconnects from the network node,
2. waits 500 ms,
3. tries to connect to network node again and check its firmware status.

2.3.4 FW update push/poll format

FW update push					
Update offer/request-Firmware data	type == 0 (1 byte)	HW version (4 bytes)	FW version (4 bytes)	FW checksum (4 bytes)	size (4 bytes)
Firmware data chunk	type == 1 (1 byte)	offset (4 bytes)	data (max 32 bytes)		

FW update poll			
Firmware buffer request	type == 1 (1 byte)	offset (4 bytes)	size (4 bytes)
Signals	type == 0 (upload refused), 2 (upload complete), 3 (save failed) 14 (save failed, invalid checksum) (1 byte)	0 bytes	

3 APPENDIX B – BIBLIOGRAPHY

1. https://devzone.nordicsemi.com/question/55309/connection-issues-with-android-60-marshmallow-and-nexus-6/
2. http://stackoverflow.com/questions/37151579/schemes-for-streaming-data-with-ble-gatt-characteristics

3. What to keep in mind when developing your BLE Android app

4 DOCUMENT HISTORY

Table 1: Document History

Revision	Date	Description
1.1	28-March-2018	Updating BLE characteristics: proxy position, location data, persisted position, device info, anchor list.
1.0	27-March-2018	Clean up of text and formatting. Updating disclaimer.
0.3	30-January-2018	Clean up of text and formatting
0.2	18-October-2017	R2 info removed.
0.1	04-October-2017	Initial release.

5 FURTHER INFORMATION

For further information on this or any other Decawave product contact a sales representative as follows: -

Decawave Ltd,
Adelaide Chambers,
Peter Street,
Dublin 8,
D08 T6YA,
Ireland.

mailto:sales@decawave.com
http://www.decawave.com/