EEL Bluetooth Integration Guide

Interface specification for third-party applications reading EEL series electronic levels over Bluetooth Low Energy

Item	Detail
Document	EEL BLE Interface Specification
Revision	1.0
Date	25 September 2026
Applies to	EEL series BLE electronic levels
Firmware	V11
Issued by	Level Developments Ltd

1. Introduction

1.1 Purpose

This document describes everything specific to EEL levels that a software engineer needs in order to build an application that discovers an EEL level, connects to it, and reads its inclination and battery data over Bluetooth Low Energy (BLE).

1.2 Audience and assumptions

The reader is assumed to be a competent software engineer familiar with BLE concepts — advertising, GATT services and characteristics, and notifications — and with the BLE API of their chosen platform. The document is deliberately independent of any programming language, framework or operating system. Where platform behaviour matters, it is noted separately in section 9.

1.3 Scope

The EEL level acts as a BLE peripheral that continuously streams measurement data. Integration is read-only: an application subscribes to the data stream and decodes it.

2. Communication overview

Once a client connects and enables notifications, the level transmits a fixed-length eight-byte packet approximately every 100 milliseconds (10 Hz). Each packet carries the current inclination and battery level. No request or polling is required, and the client does not need to write any data to the level to receive readings.

Property	Value
Transport	Bluetooth Low Energy (GATT)
Role of level	Peripheral
Role of application	Central
Data delivery	GATT notifications
Packet length	8 bytes

Property	Value
Update rate	Approximately 10 Hz (one packet every ~100 ms)
Commands required	None — the stream starts once notifications are enabled
MTU negotiation	Not required; 8 bytes fits within the default 20-byte ATT payload

3. Discovery

3.1 Advertised name

EEL levels advertise a device name of the following form:

`EEL-213899_9F92`

- The name always begins with the prefix EEL, which should be used to filter scan results.
- The part between the hyphen and the underscore (213899 in the example) is the unit serial number and is unique.
- The trailing underscore and four characters (_9F92) are a suffix relating to the Bluetooth module serial number.

3.2 Filtering

Filter discovered devices by checking that the advertised name starts with EEL, using a case-insensitive comparison. Performing this check in application code, rather than relying on an operating-system level name filter, has proved more reliable across devices.

3.3 Identifying a unit

Use the platform's peripheral identifier to connect, not the advertised name. Note that on iOS the peripheral identifier is a value assigned by the phone, not the hardware Bluetooth address, and it differs between phones for the same level. If an application needs to recognise a specific level across phones or sessions, use the unit number from the advertised name.

3.4 Scanning recommendations

- A scan window of four to six seconds is normally sufficient to find a nearby level.
- Where the platform supports it, use the most aggressive scan mode (for example low latency mode on Android). In environments crowded with other Bluetooth devices a level can be missed with less aggressive modes.
- Even so, a level occasionally fails to appear on the first scan in congested radio environments. Offer the user an easy way to rescan rather than lengthening the timeout.
- Reporting received signal strength (RSSI) alongside each result helps the user pick the nearer of two levels if connecting to more than one device.

4. Connection

4.1 Establishing a connection

- Connect directly rather than using any background or auto-connect mode.
- A connection timeout of around 15 seconds is appropriate.
- On platforms that allow the transport to be selected, request a Low Energy transport explicitly. On Android this avoids the classic Bluetooth path and reduces the likelihood of a pairing prompt.

4.2 Pairing and bonding

Pairing is not required to read data. However, some firmware revisions request a bond on first connection, which may cause the operating system to display a pairing dialog. Accepting it once is sufficient; the dialog does not normally reappear for that level.

4.3 Multiple levels

An application can connect to more than one level at once; two simultaneous connections are known to work reliably. Each connection should be managed entirely independently, with its own subscription, state and timers

5. GATT profile

Element	UUID / value
Measurement service	49535343-FE7D-4AE5-8FA9-9FAFD205E455
Measurement characteristic	49535343-1E4D-4BD9-BA61-23C647249616
Characteristic properties	Notify, Indicate (Write also present but not used)
Client configuration descriptor	0x2902 (standard CCCD)

To start the data stream, enable notifications on the measurement characteristic by writing the standard enable value (0x0001) to its Client Characteristic Configuration Descriptor. Most BLE libraries do this automatically when a notification subscription is requested.

Integrations should subscribe to the measurement characteristic specifically. Subscribing to every notifiable characteristic the level exposes would work, but risks receiving duplicate or unrelated data.

6. Data packet format

6.1 Layout

Byte	Content	Encoding
0	Reserved	Ignore
1	Inclination, high byte	Signed 16-bit integer, big-endian, across bytes 1 and 2
2	Inclination, low byte	(see above)
3 – 6	Reserved	Ignore
7	Battery charge	Unsigned 8-bit integer, percent (0–100)

Bytes marked reserved should be ignored. Their content must not be relied upon, as it may change in future firmware.

6.2 Decoding the inclination

Bytes 1 and 2 form a signed 16-bit integer, most significant byte first (big-endian). Its value is the inclination in arc seconds multiplied by ten, giving a resolution of 0.1 arc seconds.

```
raw        = (byte1 << 8) | byte2
if raw >= 32768 then raw = raw - 65536     // two's-complement sign
arcSeconds = raw / 10.0
```

The sign correction matters. In languages that have a native signed 16-bit type, casting the combined value to that type achieves the same thing. In languages without one (JavaScript and Python, for example), the explicit adjustment above is essential; omitting it causes every negative reading to appear as a large positive one.

6.3 Decoding the battery level

Byte 7 is the remaining battery charge as a whole-number percentage. It requires no scaling: 0x64 (decimal 100) is a full battery, 0x5A (decimal 90) is 90 per cent.

6.4 Validation

Discard any packet shorter than eight bytes rather than attempting to decode it.

6.5 Worked examples

Bytes 1, 2	Combined	Signed	Inclination
0x00 0x00	0	0	0.0 arc sec
0x04 0xE6	1254	1254	+125.4 arc sec
0xFB 0x1A	64282	−1254	−125.4 arc sec
0x7F 0xFF	32767	32767	+3276.7 arc sec (encoding maximum)
0x80 0x00	32768	−32768	−3276.8 arc sec (encoding minimum)

A complete packet, with reserved bytes shown as xx:

```
xx 04 E6 xx xx xx xx 5A     →  +125.4 arc seconds, battery 90%
```

7. Interpreting the measurement

7.1 Range and sign convention

The encoding can represent from −3276.8 to +3276.7 arc seconds. This is the limit of the data format, not the measuring range of the instrument. The level has a maximum measurement and output range of ±500”

7.2 Unit conversions

The level reports arc seconds. Other common units are derived as follows:

```
degrees    = arcSeconds / 3600
radians    = arcSeconds × π / 648000
mm per m   = tan(radians) × 1000
µm per m   = mm per m × 1000
```

Use the tangent form for gradient rather than a linear approximation. The linear factor of 0.0048481 mm/m per arc second is very close for small angles, but the exact form stays correct across the whole range. As a check, +125.4 arc seconds is 0.03483 degrees and 0.608 mm/m.

7.3 Zeroing (tare)

Relative measurement is implemented in the application, not on the level. Capture the current reading as an offset, then subtract it from each subsequent reading. Always subtract in arc seconds, before any unit conversion, so the relative value is correct in every unit.

8. Robust connection handling

8.1 Detecting a lost connection

BLE stacks do not reliably or promptly report a disconnection when a level is switched off or carried out of range, which can leave an application displaying a frozen, stale reading. Do not rely on the platform's disconnection event alone.

Instead, record the time each packet arrives and run a timer that checks it periodically. Level Developments' own application checks every 500 ms and treats the connection as lost if no packet has arrived for 2000 ms — equivalent to twenty consecutive missed packets. It then performs a full disconnection and tells the user.

8.2 Clean disconnection and reconnection

When disconnecting, for whatever reason, release everything associated with the session:

- Stop the watchdog timer.
- Disable notifications on the characteristic.
- Detach every data callback or event handler that was registered for the session.
- Close the connection and release the peripheral handle.
- Reset application state, including the last-received timestamp and any tare offset.

Failing to detach data handlers is a common and subtle fault. The first connection works, but after a disconnect and reconnect the old handler remains registered alongside the new one. The stream is then processed twice and the watchdog can see stale timestamps, typically showing as an incorrect reading followed by an immediate drop — a fault that only appears on the second connection and only clears when the application is restarted.

8.3 Threading

On most platforms notification callbacks arrive on a background thread. Transfer decoded values to the user-interface thread before updating any display. Keep the work done in the callback small, since it runs ten times a second per level.

8.4 Reference sequence

```
scan (low latency, 4–6 s)
    on advertisement: keep if name starts with "EEL"

connect (LE transport, direct, 15 s timeout)
discover services
subscribe to measurement characteristic → onPacket
start watchdog (every 500 ms)

onPacket(bytes):
    lastReceived = now
    if length(bytes) < 8: return
    raw = (bytes[1] << 8) | bytes[2]
```

```
    if raw >= 32768: raw = raw - 65536
    arcSeconds = raw / 10.0
    battery    = bytes[7]
    post (arcSeconds, battery) to UI thread

watchdog:
    if now - lastReceived > 2000 ms: disconnect(); report lost

disconnect:
    stop watchdog; disable notifications; detach handlers
    close connection; reset state
```

9. Platform notes

9.1 Android

Android version	Runtime permissions needed
12 and later (API 31+)	BLUETOOTH_SCAN and BLUETOOTH_CONNECT. BLUETOOTH_SCAN can be declared with neverForLocation, since location is not derived from scan results.
11 and earlier (API 30 and below)	BLUETOOTH, BLUETOOTH_ADMIN and ACCESS_FINE_LOCATION

- Request the Low Energy transport explicitly when connecting to minimise pairing prompts.
- Declare bluetooth_le as a required hardware feature so the app is only offered to compatible devices.

9.2 iOS

- Provide an NSBluetoothAlwaysUsageDescription string in Info.plist. iOS prompts the user for permission using this text; no runtime permission code is required.
- The iOS Simulator has no Bluetooth support. All testing must be done on a physical device.
- As noted in section 3.3, peripheral identifiers on iOS are assigned per phone and do not correspond to the hardware address.

9.3 General

Field users often have both hands occupied. Applications typically benefit from keeping the screen awake while connected, and from using large numeric readouts. If the display layout depends on fixed text sizes, consider exempting the main readout from the operating system's font scaling, as enlarged system fonts have been seen to break measurement layouts.

10. Recommended integration tests

- Scan and connect to a single level; confirm readings update at roughly 10 Hz.
- Tilt the level both ways through zero and confirm negative readings decode correctly and are not shown as large positive values.
- Compare the displayed value against a reference reading from Level Developments' own application.

- Switch the level off while connected and confirm the loss is detected within about two seconds.
- Walk out of range and back; confirm loss is detected and that a fresh connection can then be made.
- Disconnect and reconnect at least five times without restarting the application, confirming correct readings each time.
- Connect two levels simultaneously and confirm each reading is independent.
- Scan in a Bluetooth-congested environment and confirm the level is found within one or two attempts.

Technical queries about this specification should be directed to Level Developments at tech@leveldevelopments.com