identium®
Creating identities for all

INTEGRATION GUIDE

ID-BGW-02

Receive BLE Gateway Data in Your Own Software

UDP, TCP and MQTT; the JSON report format with sample payloads; heartbeat and two-way commands; reading iBeacon, Eddystone and battery data; signal strength and distance; example receivers in Python and Node.js



THIS MANUAL COVERS	VERSION	PLATFORM
ID-BGW-02 BLE gateway (all versions), JSON reports	–	UDP · TCP · MQTT / MQTTS
BLE Gateway Test Kit (sample payloads, commands, receivers, gateway simulator)	1.0	Python 3.8+ · Node.js 18+

Document version 1.0 · November 2026

Identium Tech Solutions Pvt. Ltd.
Plot No. 5, First Floor, Raja Dhirsain Marg, Garhi, Sant Nagar, East of Kailash, New Delhi 110065, India
+91 97181 99516 · 011-4714 7839 · WhatsApp +91 82877 32647 · info@identium.in

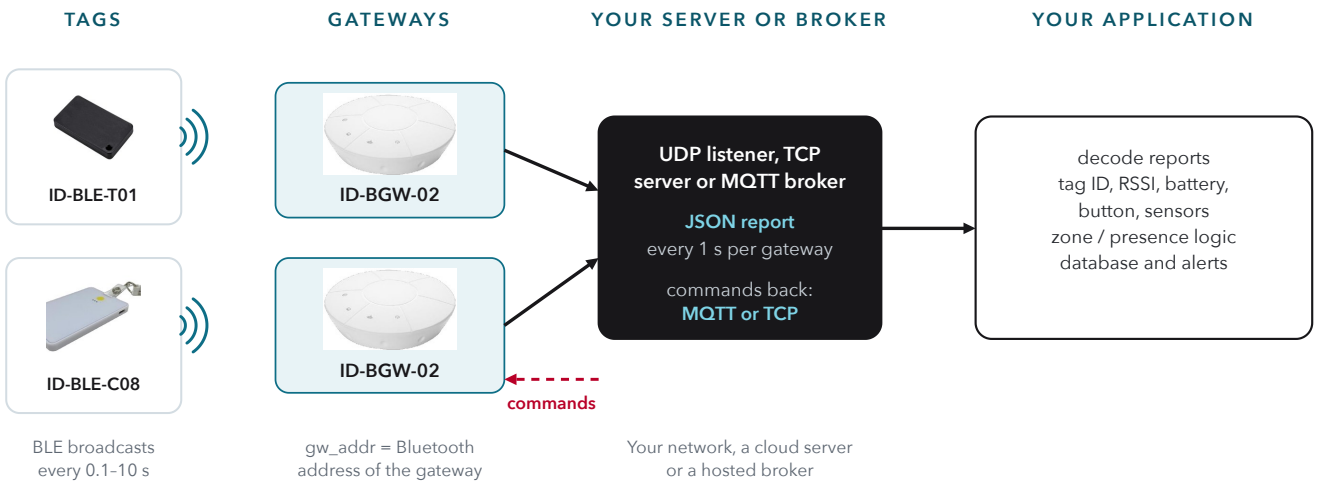
www.identiumone.com · www.indiarfidstore.com
Creating identities for all

The ID-BGW-02 gateway hears Identium BLE tags and sends what it hears to a server you choose. This guide is for the developer who builds that server: the three ways the gateway can send, the report format with every field and realistic samples, heartbeats and the commands your server can send back, how to read iBeacon, Eddystone and battery data, what signal strength can and cannot tell you about distance, and receivers in Python and Node.js that you can run as they are. Installing the gateway and its settings page are covered in the ID-BGW-02 BLE Gateway User Manual.

CONTENTS

01/ How the Gateway Sends Data
02/ Choose a Transport
03/ The Report Format
04/ More Sample Payloads
05/ Heartbeat and Keep-alive
06/ Commands: Two-Way Communication
07/ Read the Tag Data
08/ Signal Strength and Distance
09/ Example Receivers
10/ Test Without a Gateway
11/ Byte Format
12/ Troubleshooting
13/ Support

01/ How the Gateway Sends Data



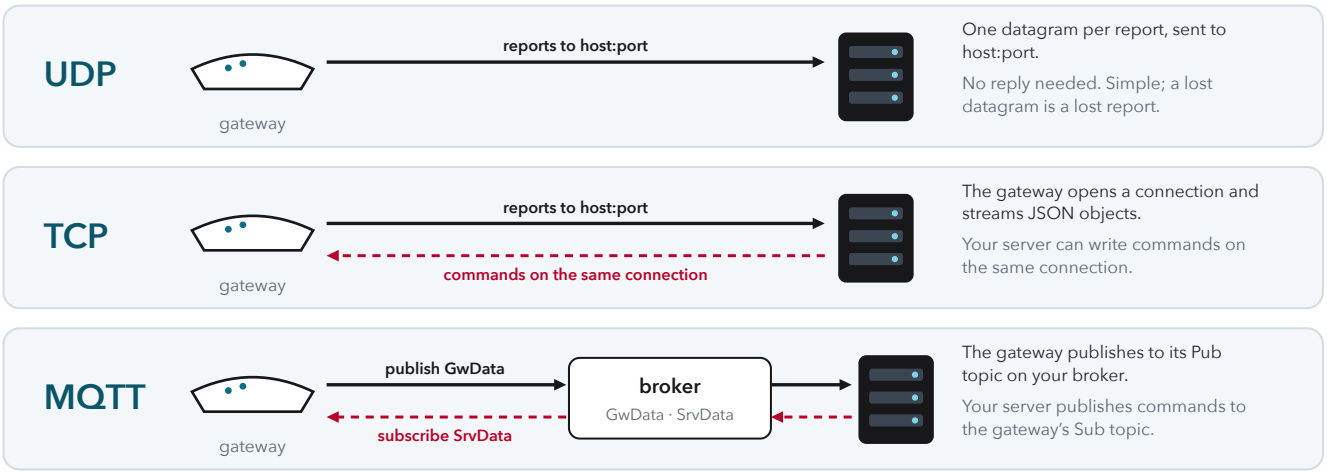
Tags broadcast; each ID-BGW-02 gateway collects what it hears and sends one JSON report per cycle to your UDP listener, TCP server or MQTT broker. Over MQTT and TCP your server can send commands back.

Who connects	The gateway is the **client**. It connects out to the host and port set on its settings page. Your server listens (UDP, TCP), or an MQTT broker sits between the two.
Transports	**UDP**, **TCP** or **MQTT**. MQTT can run over TLS (MQTTS).
Format	**JSON** (Data Mode: Json), one JSON object per message. A compact byte format also exists (section 11).
How often	One report per **Report Interval**: 1000 ms by default, 500 ms at the least. A cycle with more than 500 devices is split into several messages.
Which gateway	`gw_addr` in every message: the gateway's **Bluetooth address**, 12 hex digits without colons. The gateway's Wi-Fi address is a different number.
Sign-in	MQTT: user name, password and TLS. UDP and TCP carry no login: keep the listener on a private network or VPN, or accept only your gateways' addresses.

NOTE

The gateway's settings (server address, protocol, data mode, scan settings) are set on its built-in settings page, as described in the **ID-BGW-02 BLE Gateway User Manual**, sections 10 and 11. Most of them can also be read and changed by command (section 06).

02/ Choose a Transport



Reports go up (solid); commands come down (dashed) over TCP and MQTT.

	How messages arrive	Commands to the gateway	Good for
UDP	One JSON object per datagram. Read with a 64 KB buffer.	Use TCP or MQTT for commands.	Reports only, on a reliable network. The Identium RTLS platform listens on UDP 7628.
TCP	The gateway connects to host:port and writes JSON objects one after another with no separator. Cut the stream at the closing brace of each top-level object (the example receivers do this).	Write the command on the same connection.	Two-way work without a broker.
MQTT	The gateway publishes each message to its **Pub** topic.	Publish to the gateway's **Sub** topic.	Many gateways, cloud servers, several applications reading the same data, TLS.

MQTT topics

- Factory topics are **Pub** `GwData` and **Sub** `SrvData`. Give each gateway its own **Sub** topic so that a command reaches only that gateway, for example `site1/gw01/cmd`.
- Pub topics can be shared, because every message names its gateway in `gw_addr`, or set per gateway (for example `site1/gw01/data`) and read with a wildcard subscription such as `site1/+/data`.
- Give every gateway a different **Client ID**. Two clients with the same ID keep disconnecting each other.
- QoS 0, 1 and 2 are available on the gateway.

MQTT over TLS (MQTTS)

Turn on **SSL/TLS** in the gateway's Server Settings and use your broker's TLS port (usually 8883). For one-way authentication the gateway needs your broker's CA certificate; for two-way authentication also a client certificate and key. The certificates are packed into one file, `gw_ca.zip`, and uploaded on the gateway's settings page; Identium supplies the certificate packing tool on request.

03/ The Report Format

Every message is one JSON object with the same envelope. A **scan report** carries the tags heard in one cycle; this is payload 01 of the Test Kit, a Type 1 report with four devices:

```
{
  "pkt_type": "scan_report",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "flags": 7,
    "pkt_sn": 1532,
    "report_type": "adv_only",
    "pkt_total": 1,
    "pkt_index": 1,
    "dev_total": 4,
    "dev_num": 4,
    "dev_infos": [
      {"addr": "f0c8a0000101", "rssi": -63, "time": "2026-11-20 10:15:02", "ibcn_uuid": "fda50693-
a4e2-4fb1-afcf-c6eb07647825", "ibcn_major": 10001, "ibcn_minor": 101, "ibcn_rssi_at_1m": -59,
"adv_raw": "0201061aff4c000215fda50693a4e24fb1afcfc6eb0764782527110065c5"},
      {"addr": "f0c8a0000102", "rssi": -71, "time": "2026-11-20 10:15:02", "adv_raw":
"0201060303aafe1116aafe20000bc41b8000015180001a5e00"},
      {"addr": "f0c8a0000103", "rssi": -77, "time": "2026-11-20 10:15:02", "adv_raw":
"0201060303aafe1716aafe00ecfda50693a4e24fb1afcf0000000001030000"},
      {"addr": "4a7c19e0d4f1", "rssi": -88, "time": "2026-11-20 10:15:02", "adv_raw":
"020106020a0c0aff4c0010050b1c3d5e7f"}
    ]
  }
}
```


payloads/01-scan_report-adv_only.json (one device per line)

Envelope

Key	Type	Meaning
`pkt_type`	text	`scan_report` (tags heard), `state` (heartbeat and status), `response` (reply to a command). `command` is sent only by your server.
`gw_addr`	text	The gateway's Bluetooth address, 12 hex digits, no colons, for example `d4f98d1a2b3c`.
`time`	text	Gateway clock, `YYYY-MM-DD hh:mm:ss`, in the time zone set on the gateway.
`data`	object	The content; depends on `pkt_type`.

data of a scan report

Key	Type	Meaning
`report_type`	text	`adv_only`, `adv_srp`, `mac_rssi_only`, `stuff_card`, `ibcns_batt` or `le5_raw` (next table).
`pkt_sn`	number	Message counter, 0–65535, then starts again.
`pkt_total`, `pkt_index`	number	This cycle was split into `pkt_total` messages (1–10); this is number `pkt_index`.
`dev_total`, `dev_num`	number	Devices in the whole cycle (up to 5000) and in this message (up to 500).
`flags`	number	Gateway clock status bits.
`dev_infos`	list	One object per device heard.

Device fields (dev_infos)

Key	Type	In report types	Meaning
`addr`	text	all	Device (tag) Bluetooth address, 12 hex digits, no colons.
`rssi`	number	all	Signal strength in dBm, a negative number such as `-63`.
`time`	text	all	When the device was last heard.
`name`	text	adv_only, adv_srp, le5_raw	Device name, if the gateway found one.
`ibcn_uuid`, `ibcn_major`, `ibcn_minor`, `ibcn_rssi_at_1m`	text, numbers	adv_only, adv_srp	iBeacon fields, already decoded by the gateway. Present only when the device broadcasts iBeacon.
`adv_raw`	hex text	adv_only, adv_srp, le5_raw	The broadcast itself (section 07).
`srp_raw`	hex text	adv_srp	The scan response.
`motion`, `keys`, `batt`	numbers	stuff_card	0 still / 1 moving; button: 0 none, other values a press (SOS and other button events); battery in mV, −1 if unknown.
`ibcn_num`, `ibcn_infos`	number, list	stuff_card	iBeacons the work card itself has heard: `major`, `minor`, `rssi`.
`batt_type`, `batt_value`	numbers	ibcns_batt	Battery: type 0 = percent, 1 = mV.
`le_info`	text	le5_raw	Bit 0: Bluetooth 5 packet; bit 1: includes scan response; bit 2: Coded PHY.

Report types

Settings page	report_type	Contains
Type 1 (default)	`adv_only`	Broadcast of each device, plus decoded iBeacon fields.
Type 2	`adv_srp`	As Type 1 plus the scan response, where many tags put their name.
Type 3	`mac_rssi_only`	Address and RSSI only: the smallest reports.
Type 4	`stuff_card` (and `ibcns_batt`)	Work-card format: motion, button, battery and iBeacons heard.
Type 6	`le5_raw`	Bluetooth 5 and Bluetooth 4 broadcasts with `le_info`.

Write your receiver for these

- Branch on `pkt_type` first, then on `report_type`. Treat every device key except `addr` and `rssi` as optional.
- Normalise addresses (for example to `F0:C8:A0:00:01:01`) before you store or compare them.
- Store `time` as text or convert it with the gateway's time zone; it carries no zone. Your server's receive time is a good second timestamp.
- Handle each message on its own: a busy cycle arrives as several messages with the same layout.

04/ More Sample Payloads

The other five messages of the Test Kit. Post them to your own receiver to test it (section 10).

Type 2: with scan response and name


```
{
  "pkt_type": "scan_report",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "flags": 7,
    "pkt_sn": 1533,
    "report_type": "adv_srp",
    "pkt_total": 1,
    "pkt_index": 1,
    "dev_total": 1,
    "dev_num": 1,
    "dev_infos": [
      {"addr": "f0c8a0000101", "rssi": -62, "time": "2026-11-20 10:15:02", "name": "TAG-0101",
"ibcn_uuid": "fda50693-a4e2-4fb1-afcf-c6eb07647825", "ibcn_major": 10001, "ibcn_minor": 101,
"ibcn_rssi_at_1m": -59, "adv_raw": "0201061aff4c000215fda50693a4e24fb1afcfc6eb0764782527110065c5",
"srp_raw": "09095441472d30313031020a00"}
    ]
  }
}
```


payloads/02-scan_report-adv_srp.json

Type 3: address and RSSI only


```
{
  "pkt_type": "scan_report",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "flags": 7,
    "pkt_sn": 1534,
    "report_type": "mac_rssi_only",
    "pkt_total": 1,
    "pkt_index": 1,
    "dev_total": 3,
    "dev_num": 3,
    "dev_infos": [
      {"addr": "f0c8a0000101", "rssi": -64, "time": "2026-11-20 10:15:02"},
      {"addr": "f0c8a0000102", "rssi": -70, "time": "2026-11-20 10:15:02"},
      {"addr": "f0c8a0000103", "rssi": -78, "time": "2026-11-20 10:15:02"}
    ]
  }
}
```


payloads/03-scan_report-mac_rssi_only.json

Type 4: work-card format


```
{
  "pkt_type": "scan_report",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "flags": 7,
    "pkt_sn": 1535,
    "report_type": "stuff_card",
    "pkt_total": 1,
    "pkt_index": 1,
    "dev_total": 2,
    "dev_num": 2,
    "dev_infos": [
      {"addr": "f0c8a0000201", "rssi": -66, "time": "2026-11-20 10:15:02", "motion": 0, "keys": 0,
"batt": 3010, "ibcn_num": 0, "ibcn_infos": []},
      {"addr": "f0c8a0000202", "rssi": -58, "time": "2026-11-20 10:15:02", "motion": 1, "keys": 1,
"batt": 2890, "ibcn_num": 1, "ibcn_infos": [{"major": 10001, "minor": 101, "rssi": -61}]}
    ]
  }
}
```


payloads/04-scan_report-stuff_card.json

The second card is moving (`motion` 1), its button is pressed (`keys` 1), its battery reads 2890 mV and it has heard one iBeacon (major 10001, minor 101) at −61 dBm.

Gateway heartbeat

```
{
  "pkt_type": "state",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "state": "sta_gw_hb",
    "flags": 3,
    "ticks_cnt": 20
  }
}
```

payloads/05-state-gateway_heartbeat.json

Reply to a command

```
{
  "pkt_type": "response",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "msgId": 1001,
    "resp": "scan_filter_rssi",
    "result": true
  }
}
```

payloads/06-response-scan_filter_rssi.json

05/ Heartbeat and Keep-alive

From the gateway: heartbeat

The gateway sends a `state` message with `"state": "sta_gw_hb"` and a counter (`ticks_cnt`) at its **Bluetooth HB Interval** (BLE Configurations › Heartbeat Settings, or the `sys_hb` command in milliseconds). Match any state name that ends in `gw_hb`.

- Mark a gateway **online** when anything arrives from it: a report or a heartbeat.
- Mark it **offline** when nothing has arrived for a set time. The Identium RTLS platform uses 90 seconds; with the default 1-second report interval that is many missed cycles.

From your server: keep the link alive

With **Server HB Activation** turned on (BLE Configurations › Heartbeat Settings), the gateway restarts its connection whenever it has received nothing valid from the server for **60 seconds**. That makes a broken TCP connection or MQTT session recover by itself. Your server then has to send something at least once a minute: send `sys_srv_hb` every 30 seconds.

```
{
  "pkt_type": "command",
  "gw_addr": "d4f98d1a2b3c",
  "data": {
    "msgId": 1,
    "cmd": "sys_srv_hb"
  }
}
```

commands/sys_srv_hb.json: over TCP write it on the connection; over MQTT publish it to the Sub topic

Both example receivers do this with `--keepalive`.

06/ Commands: Two-Way Communication

Over TCP and MQTT your server can read and change the gateway's settings. A command names its gateway in `gw_addr`; the gateway answers with a `response` message carrying the same `msgId`.

```
{
  "pkt_type": "command",
  "gw_addr": "d4f98d1a2b3c",
  "data": {
    "msgId": 1001,
    "cmd": "scan_filter_rssi",
    "enable": true,
    "value": -85
  }
}
```

commands/scan_filter_rssi.json: pass only devices at −85 dBm or stronger

```
{
  "pkt_type": "response",
  "gw_addr": "d4f98d1a2b3c",
  "time": "2026-11-20 10:15:02",
  "data": {
    "msgId": 1001,
    "resp": "scan_filter_rssi",
    "result": true
  }
}
```

The reply

- `msgId`: your own number, 0–65534. Do not use 65535: over MQTT its reply goes to a fixed topic.
- `result`: `true` or `false`. On failure, `reason` says why (table below). Commands that read settings return the values in the reply.
- Send one command, wait for its reply, then send the next.

Useful commands

cmd	Keys	What it does
`sys_srv_hb`	–	Server keep-alive (section 05).
`sys_app_server`	`op` get/set, `type` UDP/TCP/MQTT, `host`, `port`, `mqtt` {`sub`, `pub`, `usr`, `pw`, `clientId`, `qos`}	Read or change the server the gateway reports to. The gateway reconnects.
`sys_mqtt_params`	`op`, `sub`, `pub`, `usr`, `pw`, `clientId`, `qos`	MQTT topics and login only.
`sys_data_mode`	`op`, `mode` byte/json	Report format. Follow it with `sys_rerun_server`.
`sys_rerun_server`	–	Reconnect to the server.
`sys_hb`	`op`, `value` (ms)	Gateway heartbeat interval.
`scan_report_interval`	`value` (ms, at least 500)	Report interval.
`scan_request_onoff`, `scan_report_rssi_only`, `scan_stuff_card_onoff`	`enable`	Report Type 2, 3 or 4.
`scan_filter_rssi`	`enable`, `value` (dBm)	Minimum signal strength.
`scan_filter_name`	`enable`, `num`, `value` [{`name`}]	Up to 8 names; `K*` matches names starting with K.
`scan_filter_ibcn_uuid`	`enable`, `value` or `num` + list	Up to 8 iBeacon UUIDs.
`scan_filter_dev_mac`	`enable`, `start`, `end`	Address range.
`scan_filter_get`, `scan_params_get`	–	Read all filters / all scan settings.
`sys_get_addr`, `sys_get_ver`, `sys_get_wifi_ver`	–	Bluetooth address and firmware versions.
`sys_gw_reboot`	–	Restart the whole gateway.
`sys_reload_def`	–	Factory settings: the gateway loses its network and server settings.
`sys_upgrade_wifi`, `sys_upgrade_ble`	`url`, `reload_default`	Firmware update from a URL (below).

Reasons for a failed command

reason	Meaning	reason	Meaning
0	Gateway address does not match	6	Operation timed out
1	Unknown command	7	Device not connected
2	Out of memory	8	Invalid state
3	Wrong parameter length	9	Execution failed
4	Parameter out of range	10	Invalid command packet
5	Parameter error	11	Unknown packet

Firmware update by command

```
{ "pkt_type": "command", "gw_addr": "d4f98d1a2b3c",
  "data": { "msgId": 1005, "cmd": "sys_upgrade_wifi",
            "url": "192.168.1.18:801/fw/wifi-image", "reload_default": false } }
```

Serve the firmware file from Identium on your own web server.

The gateway adds `&mac=` and its address to the URL, downloads the file, updates itself and reconnects; the reply (`"resp": "sys_upgrade_wifi"`, `result`) arrives after it is back. Use `sys_upgrade_ble` for the Bluetooth part. Keep `reload_default` false to keep the settings. Update one gateway first.

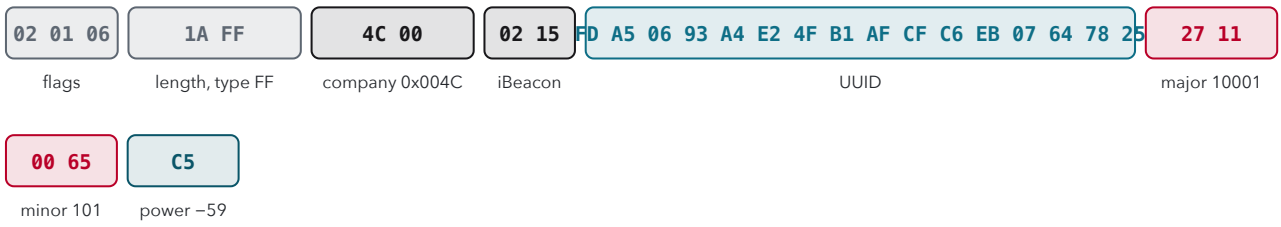
> **CAUTION**
> Commands that change the server, the data mode or the network can cut the gateway off from your server. Try them on one gateway that you can reach on site. The device connection commands (`conn_*`, used to connect to a tag and, for example, sound its buzzer) are for advanced projects: ask Identium for the details.

07/ Read the Tag Data

`adv_raw` is the tag's Bluetooth broadcast in hex (and `srp_raw` its scan response). A broadcast is a chain of records, each **length**, **type**, **data**, where the length counts the type byte and the data. Identium BLE tags broadcast in **iBeacon** and **Eddystone** formats, set per tag in the tag configuration app.

Worked example: iBeacon

adv_raw = 0201061aff4c000215fda50693a4e24fb1afcfc6eb0764782527110065c5



The iBeacon broadcast of device 1 in payload 01, split into its records.

Bytes	Read as	Value
`02 01 06`	Record type 01 (flags)	General discoverable, BLE only
`1A FF`	Record of 26 bytes, type FF (manufacturer data)	–
`4C 00`	Company ID, little-endian	0x004C: the iBeacon format
`02 15`	iBeacon type and length	21 bytes follow
`FD A5 … 78 25`	UUID, 16 bytes	`fda50693-a4e2-4fb1-afcf-c6eb07647825`
`27 11`	Major, big-endian	0x2711 = **10001**
`00 65`	Minor, big-endian	0x0065 = **101**
`C5`	Measured power at 1 m, signed	197 − 256 = **−59 dBm**

For iBeacon broadcasts the gateway also sends these values decoded, as ibcn_uuid, ibcn_major, ibcn_minor and ibcn_rssi_at_1m.

Worked example: Eddystone TLM (battery and temperature)

Device 2 of payload 01: `0201060303aafe1116aafe20000bc41b8000015180001a5e00`. The record `03 03 AA FE` lists the Eddystone service (0xFEAA); the record `11 16 AA FE …` holds the frame. Frame type `20` is TLM:

Bytes	Field	Value
`20 00`	Frame type TLM, version 0	–
`0B C4`	Battery voltage, mV	0x0BC4 = **3012 mV**
`1B 80`	Temperature, signed, 1/256 °C	0x1B80 / 256 = **27.5 °C**
`00 01 51 80`	Broadcasts since power-up	86400
`00 1A 5E 00`	Time since power-up, 0.1 s units	1728000 = 172800 s (2 days)

Frame type `00` is **UID** (10-byte namespace and 6-byte instance, device 3 of payload 01) and `10` is **URL**.

Battery

- **Eddystone TLM** frames carry the battery voltage in mV (above).
- **Work-card reports** (Type 4) carry `batt` in mV; −1 means unknown. `ibcns_batt` reports carry `batt_type` (0 = percent, 1 = mV) and `batt_value`.
- Turn millivolts into a percentage per battery type. For a 3 V coin cell the Identium RTLS platform counts 3000 mV as 100 % and 2000 mV as 0 %; rechargeable and lithium-thionyl tags need their own curve.
- Work-card reports come from tag models that broadcast the work-card format; Identium tells you which of your tags do before you rely on `keys` or `batt`.

Decoder (Python)

```
# Identium BLE Gateway Test Kit: decode a tag's advertising bytes (adv_raw / srp_raw).
# Standard Bluetooth formats only: iBeacon, Eddystone UID / URL / TLM, device name.


def ad_structures(hex_str):
    """Split advertising bytes into (type, data) pairs: length byte, type byte, data."""
    b = bytes.fromhex(hex_str or '')
    i, out = 0, []
    while i < len(b):
        n = b[i]
        if n == 0 or i + 1 + n > len(b):
            break
        out.append((b[i + 1], b[i + 2:i + 1 + n]))
        i += 1 + n
    return out


def s8(v):
    return v - 256 if v > 127 else v


def decode(adv_hex, srp_hex=''):
    """Return a dict with whatever the bytes contain: name, ibeacon, eddystone, company."""
    info = {}
    for t, d in ad_structures(adv_hex) + ad_structures(srp_hex):
        if t in (0x08, 0x09):                                   # short / complete local name
            info['name'] = d.decode('utf-8', 'replace')
        elif t == 0xFF and len(d) >= 2:                         # manufacturer data
            company = int.from_bytes(d[0:2], 'little')
            if company == 0x004C and len(d) >= 25 and d[2] == 0x02 and d[3] == 0x15:
                u = d[4:20].hex()
                info['ibeacon'] = {
                    'uuid': f'{u[0:8]}-{u[8:12]}-{u[12:16]}-{u[16:20]}-{u[20:32]}',
                    'major': int.from_bytes(d[20:22], 'big'),
                    'minor': int.from_bytes(d[22:24], 'big'),
                    'power_1m': s8(d[24]),
                }
            else:
                info['company'] = f'0x{company:04X}'
        elif t == 0x16 and len(d) >= 3 and d[0:2] == b'\xaa\xfe':  # Eddystone service data
            f = d[2:]
            if f[0] == 0x00 and len(f) >= 18:
                info['eddystone'] = {'frame': 'UID', 'tx_0m': s8(f[1]),
                                     'namespace': f[2:12].hex(), 'instance': f[12:18].hex()}
            elif f[0] == 0x10 and len(f) >= 3:
                prefix = ['http://www.', 'https://www.', 'http://', 'https://'][f[2]] if f[2] < 4 else
''
                info['eddystone'] = {'frame': 'URL', 'tx_0m': s8(f[1]),
                                     'url': prefix + f[3:].decode('ascii', 'replace')}
            elif f[0] == 0x20 and len(f) >= 14:
                temp = int.from_bytes(f[4:6], 'big', signed=True) / 256
                info['eddystone'] = {'frame': 'TLM', 'battery_mv': int.from_bytes(f[2:4], 'big'),
                                     'temp_c': round(temp, 2), 'adv_count': int.from_bytes(f[6:10],
'big'),
                                     'uptime_s': int.from_bytes(f[10:14], 'big') / 10}
    return info
```

```
def distance_m(rssi, a=59, n=2.5):
    """Rough distance from RSSI: d = 10 ^ ((|RSSI| - A) / (10 * n)). A = |RSSI| at 1 m."""
    return round(10 ** ((abs(rssi) - a) / (10 * n)), 1)
```

receivers/python/beacon.py. receivers/node/beacon.js is the same in JavaScript.

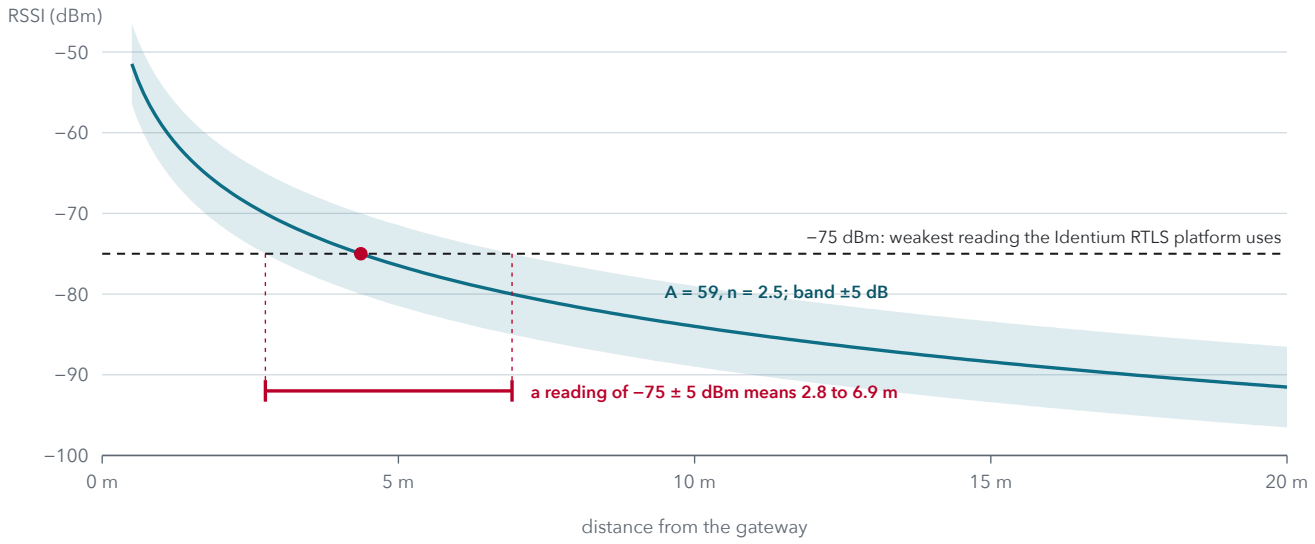
08/ Signal Strength and Distance

RSSI falls as a tag moves away from a gateway, but it also falls through walls, metal, people and with the way the tag is turned, and it changes by several dB from one reading to the next when nothing moves. Treat it as a measure of "near" and "far", not as a tape measure.

```
distance (m) = 10 ^ ( ( |RSSI| - A ) / ( 10 × n ) )

A = |RSSI| measured with the tag 1 m from the gateway   (typical 55–65)
n = path-loss exponent: about 2 in open space, 2.5–3 or more indoors
```

The model the Identium RTLS platform uses (defaults A = 59, n = 2.5)



RSSI against distance for A = 59 and n = 2.5. The shaded band is ±5 dB, an ordinary reading-to-reading swing: a reading of −75 dBm can mean anything from about 2.8 m to 6.9 m.

Calibrate per site

1. Hold a tag 1 m from the gateway for 10 seconds and average the RSSI: **A** is that value without the minus sign.
2. Move the tag to 6 m, average again, and choose **n** so that the formula gives 6 m: n = (|RSSI| − A) / (10 × log10 6).
3. Repeat for a gateway in a different kind of space (open floor, racking, corridor); each can have its own A and n.

Worked example

Gateway	RSSI of tag F0:C8:A0:00:01:01	Distance with A = 59, n = 2.5
GW-01	−63 dBm	about 1.4 m
GW-02	−71 dBm	about 3.0 m
GW-03	−77 dBm	about 5.2 m

The tag is clearly nearest GW-01: report it in GW-01's zone. The metre values are estimates, each uncertain by a factor of 1.5–2.

Use zones, not metres

- **Smooth**: average each tag's RSSI per gateway over 3–5 seconds (or turn on **Report the MAX RSSI** and use each cycle's strongest value).
- **Nearest gateway**: the tag is in the zone of the gateway with the strongest averaged RSSI.
- **Hysteresis**: move a tag to a new zone only when the new gateway is stronger by about 6 dB for two or more cycles, so a tag on a boundary does not flicker.
- **Presence**: a tag is in a zone while it is heard there above your threshold, and leaves after it has not been heard there for a set time (for example 10–30 s, several times the tag's broadcast interval).
- **Position on a floor**: use three or more gateways and a weighted estimate, and expect zone-level results, around 10 m. For sub-metre positions use ID-AOA-GW25 AoA anchors with AoA tags.

09/ Example Receivers

The Test Kit has a full receiver in Python and in Node.js (`receivers/python/receiver.py`, `receivers/node/receiver.js`). Both listen by UDP, TCP or MQTT, decode iBeacon, Eddystone and work-card data, answer with the keep-alive, can send a command, and print the same lines for the same input. The two smallest useful versions:

Python: MQTT

Needs Python 3.8 or later and paho-mqtt 2: `pip install paho-mqtt`, then `python minimal_mqtt.py` in `receivers/python`.

```
# Smallest useful MQTT receiver for ID-BGW-02 reports (Python 3.8+, pip install paho-mqtt)
import json
import paho.mqtt.client as mqtt
from beacon import decode

BROKER, PUB, SUB = 'localhost', 'GwData', 'SrvData'   # your broker; the gateway's Pub and Sub topics

def on_connect(client, userdata, flags, rc, props=None):
    client.subscribe(PUB)

def on_message(client, userdata, m):
    msg = json.loads(m.payload)
    if msg.get('pkt_type') != 'scan_report':
        return                                        # heartbeats and command replies
    gw = msg['gw_addr']                               # the gateway's Bluetooth address
    for d in msg['data'].get('dev_infos', []):
        info = decode(d.get('adv_raw', ''), d.get('srp_raw', ''))
        print(gw, d['addr'], d['rssi'], info)         # save to your database here

c = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)     # paho-mqtt 2.x
c.on_connect, c.on_message = on_connect, on_message
c.connect(BROKER, 1883)
c.loop_forever()
```

receivers/python/minimal_mqtt.py

Node.js: UDP

Needs Node.js 18 or later and no packages: `node minimal-udp.js` in `receivers/node`.

```
// Smallest useful UDP receiver for ID-BGW-02 reports (Node.js 18+, no packages needed)
const dgram = require('dgram');
const { decode } = require('./beacon');

const sock = dgram.createSocket('udp4');
sock.on('message', (buf, from) => {
  let msg;
  try { msg = JSON.parse(buf.toString('utf8')); } catch { return; }   // byte-mode frames are not JSON
  if (msg.pkt_type !== 'scan_report') return;                          // heartbeats and command
replies
  for (const d of msg.data.dev_infos || []) {
    const info = decode(d.adv_raw, d.srp_raw);
    console.log(msg.gw_addr, d.addr, d.rssi, JSON.stringify(info));   // save to your database here
  }
});
sock.bind(7628, () => console.log('listening on UDP 7628'));
```

receivers/node/minimal-udp.js

Full receivers

```
python receiver.py udp  --port 7628
python receiver.py tcp  --port 7628 --keepalive
python receiver.py mqtt --host broker.example.com --pub GwData --sub SrvData --keepalive
python receiver.py mqtt --host broker.example.com --port 8883 --tls --cafile ca.pem --user gw --
password ***

node receiver.js udp --port 7628          # the same options in Node.js (npm install first, for MQTT)
```

Options: --port, --host, --pub, --sub, --user, --password, --tls, --cafile, --keepalive, --command FILE

```
[D4:F9:8D:1A:2B:3C] scan_report adv_only packet 1/1, 4 device(s)
  F0:C8:A0:00:01:01  rssi -63  iBeacon fda50693-a4e2-4fb1-afcf-c6eb07647825 major 10001 minor 101
power -59
  F0:C8:A0:00:01:02  rssi -71  Eddystone TLM battery 3012 mV temp 27.5 C
  F0:C8:A0:00:01:03  rssi -77  Eddystone UID fda50693a4e24fb1afcf 000000000103
  4A:7C:19:E0:D4:F1  rssi -88  company 0x004C
[D4:F9:8D:1A:2B:3C] scan_report stuff_card packet 1/1, 2 device(s)
  F0:C8:A0:00:02:02  rssi -58  card motion 1 keys 1 battery 2890 mV | heard 10001/101 at -61
[D4:F9:8D:1A:2B:3C] heartbeat ticks 20
[D4:F9:8D:1A:2B:3C] response scan_filter_rssi msgId 1001 result true
```

What both receivers print for the sample payloads (shortened)

10/ Test Without a Gateway

`tools/gateway-sim.py` in the Test Kit sends the sample payloads exactly as a gateway does: one message per report, by UDP, over a TCP connection, or published to an MQTT broker. Use it to test your own receiver before a gateway is installed.

1. Start your receiver, or one of the examples, on a computer the gateways will be able to reach.
2. In `tools`, send the payloads to it:

```
python gateway-sim.py udp  127.0.0.1 7628 ../payloads/*.json
python gateway-sim.py tcp  127.0.0.1 7628 ../payloads/*.json --listen 5
python gateway-sim.py mqtt localhost 1883 ../payloads/*.json --pub GwData --sub SrvData --listen 35
```

3. Check that every message is decoded. With `--listen`, the simulator stays connected and prints any command your server sends back, for example the keep-alive after 30 seconds:

```
published 6 report(s) to GwData
command from server: {"pkt_type":"command","gw_addr":"d4f98d1a2b3c","data":
{"msgId":1,"cmd":"sys_srv_hb"}}
```

4. Run the same test from another computer, then set the gateway's Server Settings to the same host, port and protocol and click **Activate**.

TIP

For MQTT you need a broker: any MQTT 3.1.1 broker works, on your own server or as a cloud service. A firewall on the server can let local traffic through and still block the gateways: if the simulator works on the server but not from another computer, open the port.

11 / Byte Format

With **Data Mode: Byte** the gateway sends a compact binary frame instead of JSON (Timestamp and CRC16 off). JSON is easier to work with and is recommended; this is the layout for systems that need the byte format.

Offset	Bytes	Field
0	2	Header `AD BA`
2	2	Version `00 02`
4	1	Packet type: `01` broadcast, `02` broadcast + scan response, `03` address and RSSI only
5	2	Length
7	1	Clock: bit 2 = time set, bit 0 = daily restart on
8	1	Split: high 4 bits = packets in this cycle, low 4 bits = this packet (from 0)
9	2	Packet counter, 0–65535
11	6	Gateway Bluetooth address
17	2	Number of devices (up to 500)
19	…	Per device: address (6), RSSI (1, signed, strongest in the cycle), broadcast (31, padded with 00; not in type 03), scan response (31, type 02 only)
end	2	Check code: XOR of the bytes from the length field to the last data byte, written twice

12/ Troubleshooting

Problem	What to check
Nothing arrives	On the gateway: Server Settings (protocol, host, port) and **Activate**; Running Status › Application Server. On the server: the listener runs and the firewall lets that port in (UDP or TCP). A gateway on 4G needs a host it can reach over the internet.
Messages look like binary	Data Mode is Byte. Set **Json** in Report Options and click Activate.
JSON errors over TCP	One read can hold several objects, or part of one. Cut the stream by brace depth, as the example receivers do.
UDP reports cut short	Read with a 64 KB buffer. Reduce report size with filters or Report Type 3.
MQTT: no messages	Your subscription must match the gateway's **Pub** topic exactly (topics are case-sensitive). Check the broker login and, with TLS, the certificate and port. Give every gateway its own Client ID.
Commands are ignored	Publish to the gateway's **Sub** topic, not its Pub topic. `gw_addr` must be the gateway's Bluetooth address (reason 0 otherwise). Use msgId 0–65534. Send commands over TCP or MQTT.
Reply with reason 4 or 5	A value is out of range or wrong, for example a report interval under 500 ms.
The connection restarts every minute	Server HB Activation is on but your server sends nothing back. Send `sys_srv_hb` every 30 seconds, or turn the option off.
The same tag appears twice in a cycle	A large cycle is split into several messages; also check that the tag is not heard by two gateways (different `gw_addr`).
No device names	Names often travel in the scan response: use Report Type 2.
No battery value	The tag sends no Eddystone TLM frame, or is not in the work-card format. Check the tag's broadcast settings.
The gateway address is not the one on the label	`gw_addr` is the Bluetooth address; the Wi-Fi address on the label or router is a different number. Running Status › Device Address shows both.
Times are wrong	Set the gateway's time zone (NTP Settings) and use your server's receive time as well.

13/ Support

For help with installation, integration or a fault that the steps above do not clear, contact Identium. Please keep the product model, the software version shown in the app and a short description (or screenshot) of the problem ready.

Company	Identium Tech Solutions Pvt. Ltd.
Address	Plot No. 5, First Floor, Raja Dhirsain Marg, Garhi, Sant Nagar, East of Kailash, New Delhi 110065, India
Phone	+91 97181 99516 · 011-4714 7839
WhatsApp	+91 82877 32647
Email	info@identium.in
Software and documents	www.identiumone.com/resourcehub/
Buy hardware and tags	www.indiarfidstore.com

This product and its software are updated from time to time; screens and specifications may change without notice. Check the Resource Hub for the latest version of this manual.