Dronetag Scout - Sensor Format Specification

Version: 2.5

Changelog

Version	Scout	Changes
1.0	-	Initial version Added JSON+ODID data format
1.1	3.8.0+	Added heartbeat format Added "sn" field into JSON+ODID
2.0	3.10.0+	Support for MQTTS and MQTTS+WS (no client certs) BREAKING: Change URL format for MQTT Add "mac", "counter" and "recv_id" to JSON+ODID message
2.1	3.16+	Add gnss_altitude to heartbeat
2.2	3.20+	Add Avionics "tech" types to JSON+ODID messages
2.3	3.23.2+	Add FLARM framePrefix as "FL" to JSON+ODID messages Report noise_floor in decibels with Wi-Fi packets
2.4	4.0.0+	Added LTE signal strength and quality to status message Added forwarding failures Added JSON+DRI format
2.5	upcoming	The JSON+ODID message is now specified by a versioned, machine-readable JSON Schema generated from the sensor's typed data models; download scout-json-odid.schema-v1.0.json from the help page Added "registration" to JSON+ODID: drone maker/model resolved offline from the Remote ID serial (Scout Sensor+ with a valid license) Documented aviation identifiers: ICAO 24-bit address sent with IDType=2 (ADS-B) or IDType=5 (OGN/FLARM/ADS-L); flight number/callsign in SelfID Bluetooth legacy (B4) messages are aggregated and forwarded as PACKED messages (since ScoutOS 2026.05.29) Transport options rewritten to match the Sensor configuration UI; the URL-encoded configuration format is no longer available Document renamed from "SCOUT datasheet" to "Scout Sensor Format Specification"

Transport options

Messages are sent either to Dronetag's servers - the permanent Dronetag Cloud forwarder and optionally a Dronetag On-Premise instance - or to custom JSON forwarders intended for user integrations. A Scout runs multiple forwarders in parallel; a forwarder that cannot keep up does not affect the others.

Forwarders are configured in the Scout web UI (Sensor configuration). All connection options are set in dedicated fields; the URL-encoded configuration format of previous document versions is no longer available in the UI.

Basic settings

- **Enabled:** turn the forwarder on or off; when disabled no data is sent.
- **URL:** the destination server. TLS and WebSockets are selected with dedicated options, not guessed from the URL.
- **Sources:** which data to forward - drones (detected drone telemetry), aviation (manned aircraft from the aviation modules) and status (Scout health and position). Drones and aviation respect the global radius/altitude filters.
- **Client Timeout:** how long to wait for the server before the connection is considered broken (default 15 s).

Batching

Batching groups multiple messages before sending, reducing network traffic and server load at the cost of a slight delay. Without batching every message is sent separately as a single JSON line (JSONL, terminated by "\n", hex 0x0A).

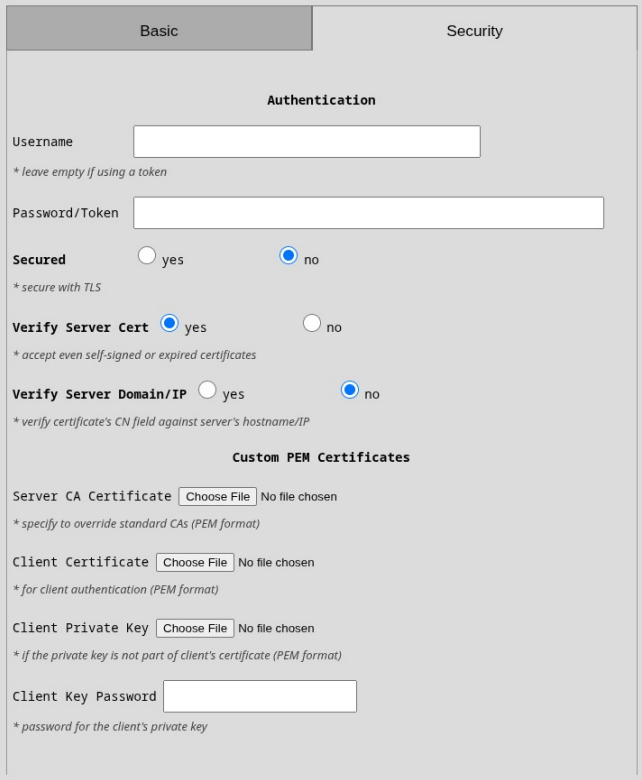
- **Enable Batching:** turn message batching on or off.
- **Batch as JSON Array:** group messages as [msg1, msg2, ...]; when disabled, messages are joined with the Items Separator.
- **Items Separator:** character(s) between batched messages: nothing, "\r\n" or "\n".
- **Batch Size:** maximum number of messages in a batch; 0 disables.

- **Batch Timeout:** maximum seconds to wait before sending an incomplete batch (e.g. 0.1 sends every 100 ms); 0 disables. With Batch Size 100 and Batch Timeout 0.1 s, a batch is sent when 100 messages accumulate or 0.1 s passes, whichever comes first.

Compression

Compression reduces bandwidth usage with gzip. Over HTTP the data is compressed only when it makes sense and the compression is indicated by the Content-Encoding: gzip header; over MQTT the data is always compressed because there is no way to hint compression.

Security settings



- **Secured:** enable TLS for the connection (HTTPS/MQTTS).
- **Verify Server Cert:** check that the server certificate is valid and not expired; disable only for testing with self-signed certificates.
- **Verify Server Domain/IP:** check that the certificate hostname (CN) matches the server.
- **Server CA Certificate:** custom CA in PEM format; it must be the top-level CA of the server's chain, not an intermediate one.
- **Client Certificate / Key:** client certificate, private key and optional key password for mutual TLS (PEM format).
- **Username and Password/Token:** HTTP Basic Auth or MQTT credentials; a token can be used as the password. Dynamic authentication (e.g. OAuth2) is not supported.

HTTP(S)

The URL may contain a full path and all data is then sent to that path. Optionally, aviation and status data can be sent to different paths. All paths must be absolute.

- **Aviation Path:** override the URL path for aviation data; empty uses the URL path.
- **Status Path:** override the URL path for status/heartbeat data; empty uses the URL path.
- **Keep Alive:** seconds between keep-alive pings when no data is being sent; 0 disables. Helps detect broken connections earlier.
- **HTTP Method:** POST (standard, recommended) or PUT.

MQTT

Specify the port in the URL, otherwise 1883 is used; for plain MQTT(S) host:port is sufficient. When connecting over WebSockets a path can also be specified (default /).

- **MQTT over WebSockets:** connect to the broker over WebSockets instead of native MQTT - useful when only HTTP/WebSocket connections are allowed (e.g. behind restrictive firewalls).
- **Drones Topic:** MQTT topic for drone detections. Templates: {sn} for the full serial, {sn4} for its last 4 digits, e.g. myorg/receivers/scout1/drones.
- **Aviation Topic:** separate topic for aviation data; empty falls back to the Drones Topic.
- **Status Topic:** separate topic for status/heartbeat; empty falls back to the Drones Topic.

Data Formats

JSON+ODID

Textual JSON format with raw fields parsed from ODID [1][2]. No indentation, nullified invalid fields (instead of passing invalid numeric values - e.g. for Location.Direction you will never see 361 that denotes invalid value in [1]).

The authoritative, machine-readable specification of this message is a versioned JSON Schema (draft 2020-12) generated from the sensor's typed data models - every field carries a description with units, valid ranges and null semantics. Download it from the help page: scout-json-odid.schema-v1.0.json. The tables below remain a human-readable overview of the same format.

Note: Bluetooth legacy (B4) transmitters send each message type (e.g. System, Location) separately. Since ScoutOS 2026.05.29 the Scout aggregates B4 messages until a valid Location message arrives and forwards the whole aggregate as a PACKED message; the aggregate is re-sent with every updated Location because the other fields rarely change. Aviation messages (AB, AL, UT, FL) always arrive as msg_type PACKED.

Key	Type	Example	Description
sn	str	1000033	
mac	str		MAC in stringified format XX:YY:ZZ:AA:BB:CC
counter	int	13	
rssi	int	-57	Reported RSSI by the receiving module
tech	str[2]	B5	Receiving technology: - "B4" (Bluetooth legacy), - "B5" (Bluetooth LE), - "WN" (Wi-Fi Nan), - "WB" (Wi-Fi Beacon), - "AB" (ADS-B 1090MHz), - "AL" (ADS-L 868MHz), - "UT" (UAT 978MHz) - "FL" (FLARM 868MHz) - "OG" (OGN 868MHz)
recv_id	int		Module type ID (SRM RID)
module_id	int	2	Module number that received the message. Corresponds to the antenna position on the box.
module_type	int	11	Module type that received the message. Mainly for internal use.
msg_type	int	15	ODID message type as defined in the reference: BASIC_ID (0), LOCATION (1), AUTH (2), SELF_ID (3), SYSTEM (4), OPERATOR_ID (5), PACKED (15), INVALID (255); we don't forward AUTH messages
noise_floor	int | null	-98	Measured noise floor from Wi-Fi receivers
registration	object | null	-	Drone maker/model resolved offline from the Remote ID serial number. Only available on Scout Sensor+ with a valid license; null otherwise. Fields: maker, model, type (uav | rid_module), certain.
odid	dict	-	see table 1.1

table 1.0 - JSON+ODID top-level fields

ODID/JSON "odid" field

BasicID	list of Basic IDs	see table 1.2
Location	dict | null	see table 1.3
SelfID	dict | null	see table 1.4
System	dict | null	see table 1.5
OperatorID	dict | null	see table 1.6

table 1.1 - JSON+ODID "odid" field

JSON+ODID "odid. BasicID" field

This field is AN ARRAY of the objects described in the table below.

For aviation data the BasicID contains the ICAO hex address with IDType=2 (ADS-B) or IDType=5 (OGN/FLARM/ADS-L; ICAO-address extension, reserved in ASTM F3411); UAT uses IDType=0. More aviation identifiers such as Flight plan or ICAO matriculation are available in SelfID.

Key	Type	Example	Description
UAType	int	0	UNKNOWN=0, AEROPLANE=1, HELICOPTER_OR_MULTIROTOR=2, GYROPLANE=3, HYBRID_LIFT=4, ORNITHOPTER=5, GLIDER=6, KITE=7, FREE_BALLOON=8, CAPTIVE_BALLOON=9, AIRSHIP=10, FREE_FALL_PARACHUTE=11, ROCKET=12, TETHERED_POWERED_AIRCRAFT=13, GROUND_OBSTACLE=14, OTHER=15
IDType	int	1	NONE=0, SERIAL_NUMBER=1, CAA_REGISTRATION_ID=2, UTM_ASSIGNED_UUID=3, SPECIFIC_SESSION_ID=4, ICAO_ADDRESS=5 (Dronetag aviation extension)
UASID	str		e.g.: 1596F350457791312042

table 1.2 - JSON+ODID "odid.BasicID[]" field

JSON+ODID "odid.Location" field

Position of the aircraft. This will be the most common message over B4. Other techs use mainly PACKED messages with location included. Used units: M - meters, M/S - meters per second, NM nautical miles (1.852 km).

Key	Type	Example	Description

Status	int	1	UNDECLARED = 0, GROUND = 1, AIRBORNE = 2, EMERGENCY = 3, REMOTE_ID_SYSTEM_FAILURE = 4,
Longitude	float | null	14.4666903	-180 - +180; 7 decimal places
Latitude	float | null	50.0739989	-90 - +90; 7 decimal places
Direction	int | null	180	0-360 degrees
SpeedHorizontal	float | null	254.25	0.0-255.0 m/s. Positive only. Invalid, if speed is >= 254.25 m/s: 254.25m/s
SpeedVertical	float | null		m/s. Invalid, No Value, or Unknown: 63m/s. If speed is >= 62m/s: 62m/s
AltitudeBaro	float | null	154.0	meter (Ref 29.92 inHg, 1013.24 mb)
AltitudeGeo	float | null	272.0	meter (WGS84-HAE)
HeightType	int	0	OVER_TAKEOFF = 0, OVER_GROUND = 1
Height	float | null		meters; can be negative
HorizAccuracy	int	12	UNKNOWN=0, 10NM=1, 4NM=2, 2NM=3, 1NM=4, 0_5NM=5, 0_3NM=6, 0_1NM=7, 0_05NM=8, 30M=9, 10M=10, 3M=11, 1M=12
VertAccuracy	int	3	UNKNOWN=0, 150M=1, 45M=2, 25M=3, 10M=4, 3M=5, 1M=6
BaroAccuracy	int	3	the same as VertAccuracy
SpeedAccuracy	int	3	UNKNOWN=0, 10M/S=1, 3M/S =2, 1M/S=3, 0.3M/S =4
TSAccuracy	int	1	UNKNOWN=0, 0.1s=1, 0.2s=2, 0.3s=3, 0.4s=4, 0.5s=5, 0.6s=6, 0.7s=7, 0.8s=8, 0.9s=9, 1.0s=10, 1.1s=11, 1.2s=12, 1.3s=13, 1.4s=14, 1.5s=15
Timestamp	str|null		date-time in ISO 8601 format; UTC, accuracy is given by the field above.

table 1.3 - JSON+ODID "odid.Location" field

JSON+ODID "odid.SelfID " field

The Self-ID Message is optionally sent in case the Remote Pilot wishes to declare its identity, flight purpose or both. This may serve as mitigation of a perceived threat by a neighbouring person or the public in case a UA is operating in the same close area.

For aviation - the "AB" and "UT" tech will provide the flight number (e.g. MNB9406) here.

Key	Type	Example	Description
DescType	int	0	TEXT=0, EMERGENCY=1, EXTENDED_STATUS=2

Desc	str		

table 1.4 - JSON+ODID "odid.SelfID" field

JSON+ODID "odid.System " field

Contains information about the Remote Pilot location and a swarm (if applicable).

Key	Type	Example	Description
OperatorLocationType	int	0	TAKEOFF=0, LIVE_GNSS=1, FIXED=2
ClassificationType	int	1	UNDECLARED=0, EU=1
OperatorLatitude	float | null	null	-90 - +90; 7 decimal places
OperatorLongitude	float | null	null	-180 - +180; 7 decimal places
AreaCount	int	1	quantity in a swarm; default 1
AreaRadius	int	250	meters, farthest horizontal distance from any UA's position in a group
AreaCeiling	float | null	100	meters, can be negative, maximal altitude of a swarm
AreaFloor	float | null	-1000	meters, can be negative, minimal altitude of a swarm
CategoryEU	int	1	UNDECLARED=0, OPEN=1, SPECIFIC=2, CERTIFIED=3
ClassEU	int	0	UNDECLARED=0, CLASS_0=1 ... CLASS_6=7
OperatorAltitudeGeo	float | null	399.0	meters (WGS84-HAE)
Timestamp	str | null		date-time in ISO 8601 format; UTC, resolution 1 second

table 1.5 - JSON+ODID "odid.System" field

JSON+ODID "odid.OperatorID " field

UAS Operator Registration Number.

Key	Type	Example	Description
OperatorIdType	int	0	OPERATOR_ID=0
OperatorId	str		

table 1.6 - JSON+ODID "odid.OperatorID " field

Sample data


```
{
 "rssi": -92,
 "tech": "B5",
 "recv_id": 2,
 "module_id": 0,
 "module_type": 11,
 "msg_type": 15,
 "odid": {
  "BasicID": [
   {
    "UAType": 0,
```

```
      "IDType": 1,
      "UASID": "1596F359746167260079"
    }
   ],
   "Location": {
    "Status": 1,
    "Direction": null,
    "SpeedHorizontal": null,
    "SpeedVertical": null,
    "Latitude": null,
    "Longitude": null,
    "AltitudeBaro": -23.5,
    "AltitudeGeo": null,
    "HeightType": 0,
    "Height": 0.0,
    "HorizAccuracy": 0,
    "VertAccuracy": 0,
    "BaroAccuracy": 0,
    "SpeedAccuracy": 0,
    "TSAccuracy": 0,
    "Timestamp": "2025-04-06T06:22:58"
   },
   "SelfID": null,
   "System": {
    "OperatorLocationType": 0,
    "ClassificationType": 1,
    "OperatorLatitude": null,
    "OperatorLongitude": null,
    "AreaCount": 1,
    "AreaRadius": 0,
    "AreaCeiling": null,
    "AreaFloor": null,
    "CategoryEU": 1,
    "ClassEU": 0,
    "OperatorAltitudeGeo": null,
    "Timestamp": "2025-04-06T07:22:58"
   },
   "OperatorID": {
    "OperatorIdType": 0,
    "OperatorId": ""
   }
  }
}
{
 "rssi": -75,
 "tech": "B4",
 "recv_id": 2,
 "module_id": 0,
 "module_type": 11,
 "msg_type": 1,
 "odid": {
  "BasicID": [],
  "Location": {
   "Status": 1,
   "Direction": null,
```

```
    "SpeedHorizontal": null,
    "SpeedVertical": null,
    "Latitude": null,
    "Longitude": null,
    "AltitudeBaro": -22.5,
    "AltitudeGeo": null,
    "HeightType": 0,
    "Height": 0.0,
    "HorizAccuracy": 0,
    "VertAccuracy": 0,
    "BaroAccuracy": 0,
    "SpeedAccuracy": 0,
    "TSAccuracy": 0,
    "Timestamp": "2025-04-06T06:22:59"
   },
   "SelfID": null,
   "System": null,
   "OperatorID": null
  }
}
```

JSON+DRI Message Format

Top-Level Fields

Key	Type	Description
receiver_data	object	Details about the module that captured the frame (see below).
odid_payload	object	Encoded Open Drone ID information (see below).
sn	string	Serial number of the Scout.

Subfield: receiver_data

Key	Type / Example	Notes
location	object | null	Optional GNSS fix of the receiver.
receiver_type	int (0–3)	Internal receiver type identifier.
component_id	uint32	Distinguishes multiple identical chips on one board.
timestamp	uint32	Custom epoch (2021-01-01T00:00:00Z) in 0.1-second ticks.

Subfield: location

Key	Type / Example	Notes
latitude	float	Latitude in degrees as a decimal number.
longitude	float	Longitude in degrees as a decimal number.

Subfield: odid_payload

Key	Type / Example	Description
counter	uint32	Monotonic per-receiver message counter.
standard	int	0 = ASD-STAN, 1 = ASTM.
encoded_message	string (base64)	Raw binary ODID message encoded in Base64. Can be parsed by the Open Drone ID Library.
wifi_beacon_info	object | null	If present, identifies a Wi-Fi Beacon transmission. Transmission-specific metadata (see below).
wifi_nan_info	object | null	If present, identifies a Wi-Fi NAN transmission. Transmission-specific metadata (see below).
bluetooth_legacy_info	object | null	If present, identifies a Bluetooth Legacy transmission. Transmission-specific metadata (see below).
bluetooth_long_range_info	object | null	If present, identifies a Bluetooth Long Range transmission. Transmission-specific metadata (see below).

Subfield: <TECH>_info

The following fields share the same structure:

- wifi_beacon_info
- wifi_nan_info
- bluetooth_legacy_info
- bluetooth_long_range_info

Field	Wi-Fi Beacon / NAN	Bluetooth Legacy / Long Range	Description
mac	✔️	✔️	Transmitter MAC address encoded in

			Base64.
rssi	✔️	✔️	Signal level in dBm (signed integer).
channel	✔️ (Wi-Fi only)	—	Wi-Fi channel (1–165).
frequency	✔️ (Wi-Fi only)	—	0 = 2.4 GHz, 1 = 5 GHz.
noise_floor	✔️ (optional)	✔️ (optional)	Noise floor value in dBm, if reported.

Sample data


```
{
 "odid_payload": {
  "counter": 25,
  "standard": "ODID_STANDARD_ASTM",
  "bluetooth_legacy_info": {
   "mac": "q6u8vN7e",
   "rssi": -116
  },
  "encoded_message":
"8hkDAh8xNTkxMTIzNDU2NzgAAAAAAAAAAAAAABImF0IAax3aHVxFmgguC28KsAcAAEyBAABSAEg1dFFzdEdNSFA5WTcAAA
AAAAAAAAAA"
 },
 "receiver_data": {
  "timestamp": 1708485109
 },
 "sn": "D11234567812345678"
}
```


Notes

- timestamp uses a custom epoch of **2021-01-01 00:00:00 UTC** and is measured in **0.1-second ticks**.
- Only one of the transmission-specific metadata objects (wifi_beacon_info, wifi_nan_info, bluetooth_legacy_info, bluetooth_long_range_info) is typically present for a given received transmission.
- encoded_message contains the raw Open Drone ID payload encoded as base64.

Heartbeat

Heartbeat messages are sent periodically in JSON format.

Key	Type	Example	Description
sn	str	D1123456781234567 8	Serial Number
timestamp	int	0	UNIX timestamp (seconds since 1970-01-01 00:00:00)
receivers	int	2	Number of currently active receiving modules
last_observation	int | null		UNIX timestamp of the last observed message
gnss_available	bool	true	Whether GNSS is enabled on the device
gnss_position	[float,float] | null		[lat, lon] if GNSS service is available and turned on
gnss_satellites	int	12	Number of visible satelites
gnss_altitude	float | null	276.9	Geo altitude in meters
gsm	GSMStatus| null		Status of cellular connection (if available)
sensors	array<SensorStatus>		list of sensors statuses

Heartbeat - SensorStatus

Key	Type	Example	Description
id	str	RD1	module identification string (RD1, RW1 ...)
uid	int	12	system (USB) port number (traceable to antenna number)
source	str	0/dri0	distinctive source name (technology name or serial path)
state	str	ok	ok, unknown, error, starting
timestamp	float		UNIX timestamp with decimal precision
tech	str		RemoteID, ADS-L, ADS-B, OGN, UAT, FLARM
messages	int		number of received messages
filtered	int		number of filtered out messages based onrestrictions (alt/distance)
last_message_at	int		UNIX timestamp of last message
last_status_at	int		UNIX timestamp of last status message
extras	object | null		Information based on module type

Heartbeat data sample


```
{
 "sn": "D17083793ED8D3DD33",
 "timestamp": 1780310741,
 "receivers": 4,
 "last_observation": 1780310741,
 "gnss_available": true,
 "gnss_position": [
  50.073992633,
```

```
  14.466609883
],
"gnss_satellites": 6,
"gnss_altitude": 262.5,
"gsm": {
  "enabled": true,
  "state": "connected",
  "quality": 100,
  "tech": "lte"
},
"sensors": [
  {
    "id": "RW1",
    "uid": 2,
    "source": "wifi/dri_wlp1s0u1u2",
    "state": "ok",
    "timestamp": 1780310741.7995672,
    "tech": "RemoteID",
    "messages": 0,
    "filtered": 0,
    "last_message_at": 0,
    "last_status_at": 0,
    "extras": {
      "module_type": 10,
      "module_type_name": "RW1",
      "module_revision": 0
    }
  },
  {
    "id": "RD1",
    "uid": 3,
    "source": "0/dri_nrf52_0",
    "state": "ok",
    "timestamp": 1780310741.7995877,
    "tech": "RemoteID",
    "messages": 85,
    "filtered": 0,
    "last_message_at": 1780306590,
    "last_status_at": 0,
    "extras": {
      "module_type": 11,
      "module_type_name": "RD1",
      "module_revision": 1
    }
  },
  {
    "id": "RD1",
    "uid": 3,
    "source": "0/dri_esp32_0",
    "state": "ok",
    "timestamp": 1780310741.7996035,
    "tech": "RemoteID",
    "messages": 0,
    "filtered": 0,
    "last_message_at": 0,
```

```
    "last_status_at": 0,
    "extras": {
      "module_type": 11,
      "module_type_name": "RD1",
      "module_revision": 1
    }
  },
  {
    "id": "ADS-B",
    "uid": 18873,
    "source": "18873",
    "state": "ok",
    "timestamp": 1780310741.7998254,
    "tech": "ADS-B",
    "messages": 32588,
    "filtered": 32588,
    "last_message_at": 1780310741,
    "last_status_at": 0,
    "extras": {
      "export_running": false,
      "export_size": 0
    }
  }
 ]
}
```


References

[1] ODID reference implementation
https://github.com/opendroneid/opendroneid-core-c/blob/dfaf3b991a660ce28748bce0e9538564f2c4ee47/libopendroneid/opendroneid.h

[2] ASD-STAN D5 WG8 technical reference