TECHNICAL PAPER

Scanning Wi-Fi and Bluetooth for known radios

A place full of radios, and a list that still has to be readable.

Off Grid Pete LLC
31 October 2026

Abstract

Most of the radios in a place are phones, headphones, and shop equipment. The few a person came to look for are in the same scan, often under a rotating address and often with no name. A raw address list does not surface them, and a full vendor registry labels the chip rather than the product. This paper is how a Fieldwatch signature is defined, how that definition is matched, and how a separate field map reads bytes the match already accepted. The scan, the alert list, and the sit are here because they are what that definition is for. A name on a row is a pattern in an advertisement.

1. Too many radios to read as they arrive

A person walks into a place and wants to know which of the radios around them are ones they already have a reason to look at. A tracker. A camera on a pole. A drone. A body-worn camera. A mesh node. A tag. The air in that place is mostly other things: phones, headphones, thermostats, terminals, and the access points of every shop on the block. A scan of that air is a list of addresses. Many of those addresses will be different the next time the same radio transmits. Many of the rows have no name. The bytes after a company identifier are in a format that belongs to whoever built the product.

Reading that list as it scrolls is not a way to find the radio. Importing every vendor prefix from a registry is not either. A registry prefix labels the chip, and the chip is in products the person does not care about. The rows that matter get buried under the rows that are merely identified. A warning on every identified row would be the same mistake, only louder. A phone that stored every advertisement as its own line would fill up before the person had finished the block.

This paper is how Fieldwatch keeps the scan and still makes the list readable. An advertisement becomes one row per address. A signature is a record in a file, applied by one matcher, rather than a branch added to the scanner for each product. The record says which bytes have to be present. A second record on the same signature, the field map, says how to read a payload once those bytes have matched. A few payloads have a shape that map cannot express, and those have a parser in the application. The signature still decides whether the row is labeled. A label is not, by itself, an alert. An alert is a separate list a person bookmarked. A sit is a named stretch of what was heard, kept after the radios have left the live list, so the afternoon can be read later. A name on a row is a pattern in an advertisement. What follows is that record, the comparison each rule makes, the field map, and the scan, the alert, and the report around them.

> **A match is not the object**
>
> A signature names a pattern in an advertisement. It does not name the person carrying the radio, and it does not prove the radio is the product printed on the row. A camera row can be a module that shares an address prefix. A finder row can be a phone taking part in the same network as a tag. Where the stock note says to look with your eyes, that is the limit of the match. A radio that never advertises is absent. An empty list is not a clear area.

2. The job, stated narrowly

The live list is an inventory of radios this phone can hear. Wi-Fi is the access points in the scan results. Bluetooth is the advertisements the controller delivers. Fieldwatch does not associate with a network, does not open a GATT connection, and does not transmit a frame of its own. The scan itself is Android's. On Wi-Fi the application calls startScan and reads the results the system returns. On Bluetooth it registers a scan callback and is handed each advertisement.

The question the signature file answers is smaller than "what is this device." It is: do the name, the address prefix, a service UUID, a company identifier, or a prefix of the payload match a rule someone wrote down? If they do, the row gains that signature's name, class, color, notes, and, when the signature includes one, a decode of the payload. If a person has bookmarked that signature, or that exact radio, the row can also beep. Those are two different decisions, made by two different lists.

Catalog 97, the stock file shipped with Fieldwatch 1.1.37, holds 255 signatures and 6,975 rules. Most of the rules are three-byte address prefixes. Nineteen signatures also carry a decode map. Fifty carry an Extra attention sentence. Every signature has a note. One signature requires every one of its rules. The rest match if any rule hits. None of the stock signatures ask for a count of neighboring radios. The engine can do that. The file does not currently ask.

3. What was already known about these advertisements

The address on a Bluetooth advertisement is a poor identity. Phones and tags rotate it. Martin, Mayberry, Donahue, Foppe, Brown, Riggins, Rye, and Brown showed in 2017 that randomization often fails to hide a device, because other fields in the same advertisement stay stable. Becker, Li, and Starobinski showed in 2019 that anonymized Bluetooth devices can still be tracked from those fields. That is why a Fieldwatch signature prefers a name, a service UUID, or a manufacturer prefix over the address, and why a signature created from one radio warns that a randomized address may change.

Apple's Continuity advertisements are the clearest public example of a payload that is not a product name. Celosia and Cunche, in Proceedings on Privacy Enhancing Technologies in 2020, reversed the type-length-value layout Apple puts in company identifier 0x004C. A type byte says whether the advertisement is proximity pairing, Nearby, Find My, Handoff, or one of the other Continuity messages. Fieldwatch names those types. It does not recover an email address or a phone number from them, and it does not run a tracking procedure against them. The type is a label on a row the phone already heard.

Remote ID is a published standard rather than a reversed one. ASTM F3411 packs identification and location into 25-byte messages. The OpenDroneID reference code is the reading Fieldwatch follows for the heading byte. Bluetooth company identifiers and 16-bit service UUIDs are assigned numbers, published by the Bluetooth SIG. An IEEE OUI is a registry entry. None of those registries is a signature. A signature exists only after the same bytes have been seen often enough, on this kind of radio, that a rule is worth keeping. A prefix that belongs to a

chip vendor will label every product that uses the chip.

4. Two scans, and they are not the same radio

Bluetooth is a stream. BleRadio builds one empty ScanFilter, which matches every advertisement, and asks for every match with no batching delay. An empty filter is deliberate. An unfiltered scan is something Samsung will refuse while the screen is off. The callback reads the advertisement bytes through BleAdParser. The name is the name in the advertisement. The paired-device cache on this phone is not consulted, because that name is not what is on the air.

The scan mode follows the intensity setting: low latency, balanced, or low power. A failure backs off, doubling from 4 seconds up to 30. A session that has gone quiet for 18 seconds, after it has been running at least 12, is treated as suspended by the system. It rests, then starts again. A long low-latency session is recycled at about 70 seconds so the vendor does not suspend it. Balanced runs longer. The saver setting runs much longer. The rest between those cycles is a couple of seconds when the scan has been healthy, and longer when it has been failing. Hits in the last minute are counted, up to a few thousand, so the diagnostics line has something to show.

Wi-Fi is a batch. WifiRadio calls startScan and, when the results arrive, walks the list Android returns. A result is fresh only when this application asked for that scan. A spontaneous callback is often a partial or cached set, and treating it as fresh would age every other access point off the list. Android also rations scan starts. Fieldwatch keeps its own gate: four starts inside two minutes, and about 28 seconds between starts. When the system reports that throttling is off, the floor drops to 8 seconds. A refused start backs off from 8 seconds toward 45. Results that arrive more often than about three times a second are ignored.

The two radios do not share a row. The key is the kind, then the address: BLE and a MAC, or WIFI and a BSSID. The same six bytes on both radios are two devices. A Wi-Fi result that is not a fresh scan, and that matches a row already held, updates the name, the signal, and the information elements without counting a new hit. A fresh scan, and every Bluetooth advertisement, updates the row in full.

5. From the callback to a row

DeviceStore.ingestBatch is the door. The address is normalized. If the row is new, the store writes the name, the signal, the channel, the vendor from the address prefix, whether the address is randomized, the service UUIDs, the first manufacturer record, and the advertisement facts. The manufacturer hex kept on the row is capped at 512 characters. The raw advertisement is capped at 1,024. The signal history keeps the last 40 measured samples. A signal Android did not actually measure does not enter that history and does not move the minimum or the maximum.

A later advertisement merges. A blank name does not erase a name already stored. Service UUIDs accumulate. Manufacturer records and service-data records accumulate on the facts object. The vendor-IE list is merged. The row remembers that it has seen a Fast Pair pairing advertisement if any advertisement in its life set that bit. The GPS sample, when the phone has one, is where this phone was, not a position the other radio sent. An advertised aircraft position is a different pair of fields, filled later by the Remote ID parser.

Labeling is not done on every advertisement. relabel builds a fingerprint from the name, the hidden-network flag, the vendor information elements, the service UUIDs, and the manufacturer and service-data bytes. Signal, GPS, and the gone flag are not in it. If the fingerprint matches the one stored for that key, the signature search is skipped. If any signature in the file asks for a peer count, an OUI cluster, or sequential addresses, the fingerprint is not enough: a neighbor that did not change can complete the count, so the whole live list is scored.

The live list is not the log. A row with no signature is dropped after about 3 minutes of silence. A row with a signature is dropped after about 15 minutes. The steady target is about 400 rows. A burst of new addresses may pass that, up to a hard ceiling of about 900, and the unnamed rows go first. A row that already has two GPS samples and is still inside the linger window is kept ahead of a row that has none. The rotating log is the other record. It appends a line per observation, including a truncated raw field. The sit, described later, keeps neither the raw bytes nor a line per advertisement.

6. A signature is a record

A signature is one record. The file calls it a fleet. The phone calls it a signature. They are the same object. The record has an id, a display name, a class, a color, a note, an optional Extra attention sentence, a match-any flag, four peer fields, a list of rules, and an optional decode object. It also stores an enabled flag and a built-in flag. The matcher receives the list. It does not open the file looking for a product name.

The id is the stable key. fleet-rayneo stays fleet-rayneo if the display name changes. A brand-new id is a new signature, and a phone that has already saved an older catalog will not invent one. A new id needs a catalog-version bump and a migration. A new rule on an id the phone already has can arrive as an update.

The class is a label, not a test. The names on the phone are Finder tags, Retail beacons, Signage, Wearables, Surveillance, Drones, Pentest, Public safety, Vehicle, Glasses, Audio, Cameras, Thermostats, Access control, Health, Home IoT, ISP / routers, Mesh, Phones / PCs, and Other. Body-worn is an older stored value. The list shows it with Wearables, and the stored value remains so an older pack still opens. Public safety is the label stored in the file as law enforcement. A blank Extra attention sentence still shows that label. The label is the class. It is not a finding about a person.

A few classes change a later decision. None of them change the match. Drones are on the default watchlist even when Extra attention is empty, which is why Remote ID can beep on a new install with no sentence of its own. Finder tags, Retail beacons, and Wearables are the classes the moving-with-you test reads. Retail beacons are also the class allowed to sit beside the generic iBeacon label. The color is an index into a palette. It paints the row. It does not match.

Three pieces of text travel with the record, and they are three decisions. The note is the sentence on the detail page. Extra attention is the sentence that marks the live row, fills the amber card, and adds a line to the sit report. The bookmark is a different list, and it is what beeps. A person can remove the bookmark and keep the sentence, or the other way around. The stock camera sentences say the hit is a pattern and say to look. The matcher adds no confidence number on top of that wording.

The built-in flag marks a stock row. A custom row is left alone when the stock file is updated, even if the id collides. The enabled flag on the signature is kept across that update. The switch the editor shows, and the switch the matcher reads, is the one on each rule.

7. A rule is one comparison

Kind	What is written	When it hits
OUI	An address prefix in the text. The radio may be Wi-Fi, Bluetooth, or unset.	The address starts with that prefix. A three-byte prefix is indexed. Unless the rule is Bluetooth-only, the same prefix is also tried against Wi-Fi vendor elements.
MAC prefix	An address prefix in the text, often the whole address.	The address starts with it. A three-byte prefix is indexed. Vendor elements are not checked.
Name contains	A substring in the text.	The advertised name contains it, ignoring case. A blank is left out of the plan.
Name glob	A pattern in the text.	The whole name matches, ignoring case. A star is any run of characters. A question mark is any one character. No wildcard means the whole name.
Service UUID	A UUID in the text.	That UUID is listed, or it is the UUID of a service-data record. Four hex digits also match the Bluetooth base UUID. Bluetooth unless the rule names a radio.
Service data	An optional UUID in the text, and optional hex in the prefix.	A UUID and hex: the hex starts that service’s data. A UUID and no hex: any payload on that UUID. Hex and a blank UUID: the hex may sit anywhere, and the byte-reversed hex is tried too.
Manufacturer id	A company identifier.	A manufacturer record uses that number. Zero is company zero. Bluetooth unless the rule names a radio.
Manufacturer data	A company identifier and a hex prefix.	The company matches, or the company was left at zero, and the payload starts with the hex. An empty prefix is left out.
Radio kind	Wi-Fi or Bluetooth. No text.	The row is that radio. Unset matches either radio.
Hidden network	No text.	The Wi-Fi result is a hidden network.
Vendor element	A prefix in the text.	A Wi-Fi vendor element starts with it. The address is not checked. 00:50:F2 and 00:0F:AC are refused.

The three prefix kinds are easy to confuse, and they are the reason the index has three maps. An OUI rule feeds the address map and, unless it is Bluetooth-only, the vendor-element map. A MAC-prefix rule feeds the address map only. A vendor-element rule feeds that map only. A prefix of exactly three bytes, six hex characters, is the map key. Any other length is kept in a short side list and tested with a starts-with. A full address is that longer prefix. The elements 00:50:F2 and 00:0F:AC mark WPA and RSN. They are dropped from the vendor-element index and ignored when a vendor element carries them.

A name glob is anchored at both ends. Flock does not match Flockhart, and Flock- followed by a star still matches a longer name. A question mark is any one character, not a digit, so a pattern of eight question marks matches letters as well as digits. There is no regular expression. Name contains is a substring of the same name, also ignoring case. A blank name on the air does not satisfy either rule, and a blank pattern is left out of the plan.

A service UUID of four hex characters also matches the Bluetooth base UUID that contains those four characters, the form 0000xxxx-0000-1000-8000-00805F9B34FB. A 128-bit UUID matches itself. It is shortened to four characters only when it actually is that base UUID. The search looks at the UUIDs in the advertisement and at the UUID on each service-data record.

Service data is the rule with more than one shape, because a prefix and a UUID do different jobs. Both blank, and the rule is left out. A manufacturer id of zero matches a record whose company identifier is zero. It does not mean

any company. Manufacturer data is the rule where zero means any company, and the hex then has to start the payload.

The radio field scopes the rule. Hidden network and vendor element are Wi-Fi. Service UUID, service data, manufacturer id, and manufacturer data are Bluetooth unless the rule names a radio. An OUI, a MAC prefix, and a name follow the radio written on the rule, and an unset radio is tested on both. A signature is Wi-Fi-only when every rule is Wi-Fi-only, and Bluetooth-only when every rule is Bluetooth-only. The search for a Wi-Fi row does not walk the Bluetooth-only plans. A rule that is switched off stays in the record and is left out of the plan. A signature with no rule left on matches nothing.

8. What one signature can require

Match any is the default. One hitting rule is enough. On that setting an indexed prefix hit, including a vendor element that was indexed, is enough by itself. If the prefix maps missed, any one of the other rules is enough. Match all clears that shortcut. Every enabled rule has to hit. The stock file has one signature of that second kind.

RayNeo is that signature, and it is worth writing out because the shape is the point. The id is fleet-rayneo. The display name is RayNeo. The class is Glasses. Match any is off. There is no decode map. The first rule is manufacturer id 0x0BC6, which the stock note calls the TCL company identifier. The second rule is a Bluetooth name glob, RayNeo*. Both have to hit. A radio that only carries 0x0BC6 does not take the name. A radio named RayNeo without that company id does not take it either. The Extra attention sentence says the same limit and says to look. The class did not do the matching. The two rules did.

Remote ID is the other shape, and it is the one that carries a field map. The id is fleet-remote-id. The display name is Remote ID. The class is Drones, so a new install bookmarks it even though Extra attention is empty. Match any is on. Three rules, and any one is enough: service UUID FFFA, vendor element FA:0B:BC, and vendor element 6A:5C:35. The stock note says the last of those is the French Direct Remote ID element. It labels the row. Position and the operator pin still come from the ASTM messages reached by the other two rules. The decode object does not help any of the three rules succeed. It runs after they have.

One signature cannot mix the two shapes. It is every rule, or any rule. It cannot say that a name is required and that either of two prefixes will do. It cannot exclude a second name. A match-any signature that contains a loose name and a tight prefix will match on the name alone. The tight prefix does not restrain it. The restrained test is a second signature with match any turned off.

The peer fields are on the same record, and the stock file does not turn them on. They are a minimum count, a window in seconds, a flag to cluster by the address prefix, and a flag to count sequential addresses. The pass runs only when the count is above zero or either flag is set, and it runs after the conflict drops below. The window is the stored value, or 60 seconds when that value is empty. A cluster-by-prefix signature ignores a neighbor with a different prefix, and it will consider a blank or randomized address that did not itself match. Sequential addresses count when the last two bytes differ by at most 64. When the pass runs and the minimum was left at zero, the count required is 3. The stock file leaves the count at zero and both flags off, so a hit today is one radio.

9. How the library is kept

The on-disk format is fieldwatch-signatures, format version 1. A pack carries a catalog version and the list of records. Catalog 97, the stock file with Fieldwatch 1.1.37, holds 255 signatures and 6,975 rules. 6,081 of the rules are address prefixes. Nineteen signatures carry a decode map. Fifty carry an Extra attention sentence. Every signature has a note. One requires every rule. The rest match if any rule hits. None of the stock signatures ask for a count of neighboring radios. The fields are there. The file does not currently set them.

The stock list is written out as that JSON from rules authored against captures and against published layouts. A registry of every IEEE prefix, every Bluetooth company identifier, or every Fast Pair model number is not this file. A prefix that belongs to a chip vendor would label every product that uses the chip, and the rows a person came to look for would sit under the rows that were merely identified. A signature is kept when the same bytes have been seen often enough, on this kind of radio, that the rule is worth the row it will create. The Fast Pair model table that ships with the application is a list of numbers used after a signature has already matched service UUID FE2C. The table is not a signature. The UUID is.

Phones already in the field do not need a new application to gain a rule on a signature they already have. Update stock catalog downloads the published file and overlays built-in rows whose id starts with fleet-. The enabled flag this phone already saved is kept. Rules the operator added, which are not in the new file, are kept and appended. A custom signature that is not marked built-in is left alone, even if the id collides. The watchlist is not in that function, so an update does not bookmark a new row and does not remove a bookmark. A decode source the running application does not know is dropped on import. The signature still matches. It has no field map until an application that knows the source reads the pack.

On every launch, built-in rows are patched from the copy compiled into the application. A stock rule that is missing from the saved row is added. Rules already there stay. That is how a small correction reaches a phone that has not tapped the update. A brand-new fleet id still needs the catalog-version bump, because a phone that saved an older catalog will not invent a row it has never stored.

A person can also write a signature on the phone, from the radio on the screen, or import a pack exported from another phone. The suggestion starts with the full address, so that one radio stays labeled even if it has no name. It then adds a name rule, up to three service UUIDs, and a manufacturer prefix when those exist. A randomized address gets a note that the address may change and that the name, the UUID, or the manufacturer rule is the one that will see the next address. The suggestion is a draft. It is not added to the stock file. Saving it makes a custom row, with the same fields as a stock row.

10. How the matcher is compiled

SignatureEngine.compile walks the records once and builds a plan. Rules that are switched off are already gone. Three-byte OUI, MAC-prefix, and vendor-element rules go into the maps. Longer and shorter prefixes go to the side list. Every other rule becomes the small matcher for its kind. The compiled form is cached until the fleet list object changes.

On a match-any signature a map hit returns immediately. On a match-all signature the map is not a shortcut, and every remaining rule is tested against the row. A Wi-Fi address with the local bit set is tried a second way. Guest, mesh, and some vehicle radios set that bit on a burned-in prefix. The matcher clears it and looks the universal prefix up in the same Wi-Fi map. That recovery is Wi-Fi only, and only for a prefix of exactly three bytes. A Bluetooth randomized address is not turned back into a factory OUI.

The fingerprint in section 5 is what skips this work when the advertisement has not changed. A signature that asks for a peer count forces a full pass, because a neighbor that did not change can complete the count.

11. One radio, then the few conflicts the file cannot settle

The file can say that two signatures match. It cannot say that one of them should step aside. Five of those arguments are written in the engine, after the rules have run, and they name fleet ids. They are not signatures. A new product does not get a new function here unless two existing rows have started labeling the same advertisement and the file has no way to prefer the specific one.

iBeacon is a payload layout. If the radio already matched a signature whose class is not beacon, the generic iBeacon label is removed. Beacon-class rows stay beside it, which is why a Minew, an Estimote, a Kontakt, or an Atrius cart tag can show both. The Atrius advertisement does not name a store. The same pattern has been heard on carts in more than one chain. A Tesla phone-key advertisement is an Apple iBeacon with a prefix the catalog stores. That radio never takes the generic iBeacon label. It stays Tesla.

Company 0x08AA is DJI, and it is also an Osmo camera and a DJI Power station. If the Osmo row or the DJI Power row matched, the DJI row is removed. A Polar heart-rate strap advertises a GoPro service UUID along with its own. If the Polar row matched, the GoPro row is removed. A Meraki access point carries a Cisco vendor element. If the Meraki row matched, the Cisco row is removed.

Offline Finding is the last of these, and it is the one that surprises people. Company 0x004C, type 0x12, is the Find My advertisement. The stock row for that advertisement is Apple AirTags. An iPhone advertises it too, usually on a rotating address that is not the address of its other Apple advertisements, and usually with no name. The AirTags label is removed only when that same address already matched Apple Device or Apple audio, or the bytes are Apple Continuity, and the name does not contain AirTag, and service UUID FD44 is absent. Two addresses remain two rows. One phone can sit on the list as an Apple device and, separately, as AirTags. The decode text on the Find My bytes says they are used by AirTags, by Find My accessories, and by Apple devices locating themselves. The row is the advertisement. It is not a claim that the radio is an AirTag.

The peer test in section 8 runs after those drops, and only for a signature that asked for it. The stock file does not ask, so the drops above are the last word on a hit.

12. A decode map is a second record

Matching decides the label. The decode object does not get a vote. It is a second record on the same signature, and it runs only on signatures the matcher already attached, and only when that record has at least one field. The interpreter is one function. Nineteen stock signatures use it. Thirteen point it at manufacturer data. Six point it at service data. Eight fields in the stock file are marked for the live row. A manufacturer map reads a Bluetooth advertisement. A signature whose rules are all Wi-Fi has no such advertisement. The FFFA service map is the one that also reads a Wi-Fi element, after the parser below has framed that element as the same payload.

The record names a source, manufacturerData or serviceData. Any other source name is kept only long enough for import to drop the map. The signature still matches. A manufacturer map may name a company id. When it does, only records with that company are read. When it does not, every manufacturer record on the advertisement is a candidate. The map may also ask for the two-byte little-endian company id to be written in front of the payload before the offsets are counted. The stock tire-sensor row does this, because the status bit it wants lives in those two bytes. A service map names a UUID, and only that service’s data is read. Service UUID FFFA is the Remote ID exception: the Bluetooth service data is read, and any Wi-Fi vendor element that the Remote ID parser has framed

as the same FFFA payload is read with the same map. One document covers both radios.

A field is an instruction, and the order is fixed. A gate runs first. Then a slice is taken at the offset, for the field’s length or for the width of its type. The type is read in the field’s endian, little-endian unless the file says otherwise. The types are an unsigned or signed integer of 8, 16, or 32 bits, an unsigned 24-bit integer, a 32-bit float, a bit field, a UTF-8 string, hex, a six-byte MAC, or a boolean. A bit field uses a bit offset and a bit width inside that slice. A boolean is yes when any byte in the slice is non-zero. After the number is read, a remainder is taken if the field sets one, then the number is multiplied by a scale, then an offset is added. An enum table replaces that number with a word, looked up from the raw value. A unit string is appended for display. An enum note is a sentence shown when that word is the one decoded. A field whose slice runs past the end of the payload is skipped. An empty display is skipped.

A gate refuses the field before any of that. The operations are equal, not equal, mask, inverse mask, and length. Equal and not-equal compare the bytes at the gate’s offset with the hex in the gate. Mask means every bit set in the gate is set in the payload. Inverse mask means every bit set in the gate is clear. Length compares the payload size and ignores the offset. A gate may carry a second gate, and both have to pass. The second gate may itself carry another. The first payload that produces a value for a field id wins. A later payload, and a later instruction with the same id, does not overwrite it. That is why one map can hold two layouts for the same label: the layout that fails its gate leaves the id free for the layout that passes.

Govee is the stock map that needs both the length gate and the remainder. The signature itself is names only, match any, class Home IoT, with no Extra attention sentence. The map is manufacturer data, with no company id, so whichever manufacturer payload is on the advertisement is what the fields read. The sensor sometimes appends the ASCII marker INTELLI_ROCKS. That suffix is cut off before a length gate runs. One layout is gated on a payload of 7 bytes. Temperature is a signed 16-bit little-endian integer at offset 1, times 0.01, in degrees Celsius. Humidity is an unsigned 16-bit integer at offset 3, times 0.01, in percent. Battery is an unsigned byte at offset 5, in percent. The other layout is gated on a payload of 5 bytes and on a first byte equal to zero. Temperature and humidity then read the same unsigned 24-bit big-endian integer at offset 1. Temperature multiplies it by 0.0001. Humidity takes the remainder 1000 and then multiplies by 0.1. Battery is the byte at offset 4. Both layouts use the field ids temperature, humidity, and battery. The 7-byte instructions are listed first, so a 7-byte payload fills those ids and the 5-byte instructions never overwrite them. A 5-byte payload fails the length-7 gate, and the later instructions run.

Remote ID shows the same language on a published layout. The map’s source is service data for UUID FFFA. The message type is a bit field at offset 2, bit offset 4, width 4. The low four bits of that same byte are the protocol. Latitude is a signed 32-bit little-endian integer at offset 7, times 10 to the minus 7, in degrees. Its gate is equal: the byte at offset 2 has to be 0x10, which is a location message at protocol zero. The basic-id fields, including the UTF-8 identifier at offset 4 for 20 bytes, are gated on that same byte equaling zero. A location message does not fill the identifier, and a basic-id message does not fill the latitude. The heading the row shows is not taken from this map. It is computed with the location message in the parser below. The map and the parser share the payload. They do not share the heading byte.

Two flags change the live list rather than the detail page. live puts the decoded word on the row. liveEmphasis names the raw values that use the stronger chip. Every other live value uses the quiet chip. The chips do not beep. The list does not parse the map on each frame. The chips are stored on the sighting when the advertisement bytes change, and the list reads the stored chips. No fleet id is special inside the interpreter. A signature that sets the flag gets the chip. The parse is cached by the fleet id, the source, the payload hex, and a fingerprint of the fields. Fractions are formatted in the US locale, so a phone set to a language that uses a comma still shows a period.

13. The parsers that stay in code

The field map is a random-access read of a fixed layout. It cannot walk a list of typed messages, and it cannot express a test such as “the top five bits of byte 0 are 0x10, and then pool every address that shares bytes 4 through 11.” Those layouts have a parser. The parser does not decide that the radio matches. The signature file still does that. The parser runs because the radio matched, or because the bytes are a layout the detail page knows how to name.

AdvPayloadDecoder is the naming parser for a handful of published layouts. Company 0x004C is walked as Apple Continuity: a type, a length, and a value, repeating. The detail page names the type. iBeacon, AirDrop, proximity pairing, Hey Siri, AirPlay, Handoff, Nearby, and Find My each get a short reading. Company 0x0006 is the Microsoft advertisement. Company 0x0157 is AltBeacon. Service UUID FE2C is Fast Pair. A three-byte FE2C payload is a model id, and the model name comes from a table that ships with the application. The table is a registry of numbers, not a signature. The signature match is the UUID. Service UUID FEAA is Eddystone, and the Find Hub frames inside it, types 0x40 and 0x41, are named as such. Anything else in those company ids or UUIDs is left as a byte count. The decoder does not invent a layout it has not been given.

OpenDroneId reads ASTM F3411 messages. Each message is 25 bytes. The high nibble of the first byte is the type. Type 0 is the basic id. Type 1 is the location. Type 3 is the self id. Type 4 is the system message, which carries the operator position. A pack whose type nibble is 0xF carries a count and then that many messages. Bluetooth service data on UUID FFFA is read directly. A Wi-Fi vendor element with OUI FA:0B:BC and type 0x0D is the same messages in a Wi-Fi frame. The parser wraps each of those messages as a synthetic FFFA payload, with the 0x0D marker and the counter in front, so the stock Remote ID field map can run on Wi-Fi without a second copy of the map.

The heading is not taken from that map. The location message’s direction byte, plus the east-west flag, follows the OpenDroneID reference code. A direction byte of 255 is no heading. A speed byte of 255 is no speed. Latitude and longitude are a signed 32-bit integer times 10 to the minus 7. The operator latitude and longitude are stored beside the aircraft position, and they are never used as the aircraft pin. A system-message altitude of zero is absent. It is the encoded form of the -1000 meter sentinel, not a reading of sea level. The basic id is a serial number only when its id type is the serial type. A session id or a CAA registration does not get a maker name from the serial table.

Cta2063 is that table, and it is only a label on a serial the Remote ID parser already accepted. The longest prefix wins. The prefixes in the application are Freefly under 1817, BRINC under 1914, Teal under 1839, Autel at 1748C, and Skydio at 1668B. Bare 1748 and bare 1668 are not those makers. The Remote ID row stays Remote ID. The maker is a line on it.

SmartTag is the parser the field map cannot host. The signature row already matches service UUID FD5A, the SmartTag names, and company 0x0075. The state is not a match rule. The frame has to be at least 13 bytes, and the top five bits of the first byte have to be the pattern the parser requires. The low three bits are the state. States 1 and 2 read as separated from the owner. State 3 reads as separated and away more than 24 hours. States 4, 5, and 6 read as with the owner. States 0 and 7 stay unlabeled, and the rest of a frame that passed still shows. Aging is a little-endian integer in bytes 1 through 3. The privacy id is bytes 4 through 11. It is not a serial number, and it is not the row’s key. Battery and the UWB flag are byte 12. Bytes after that are ignored.

The live chip for that state is attached in withLiveDecode, next to whatever chips the field map produced. Separated uses the stronger chip. With owner uses the quiet chip. The chips do not beep. Extra attention stays off on the SmartTag row, because that flag would mark a tag that is still with its owner. When the latest frame is separated, the privacy id is copied onto the sighting and, from there, onto the sit row. A with-owner frame, an

unlabeled frame, or a frame that failed the header clears it. The sit does not keep the advertisement bytes, so the id has to be copied while the frame is in hand. The following test, below, is the only place addresses that share one privacy id are treated as one subject. The live list still shows them as separate rows, because the address is the key.

14. A label is not an alert

fleetIds on the row are the signatures that matched. The watchlist is a different list. A WatchTarget points at a fleet id, or at one device key, which is the kind and the address. Alerter.checkLive returns immediately when alerts are off or the watchlist is empty. A signature that matched, and that nobody bookmarked, changes the row and does not beep.

For each watchlist row and each radio still on the air, shouldRaiseWatch decides. A device key marked Mine is skipped. That skip covers the beep, the voice, the flash, the vibration, and the notification, for that key only. Every other radio of the same signature still alerts. A watchlist row that points at a device key alerts only when its alert flag is on. A custom name does not turn that flag on. A note pinned to the address does not turn it on either. A Bluetooth address that is randomized cannot take a custom name, because the name would stick to an address that is about to die. A Wi-Fi address with the local bit set can, because those addresses usually stay put.

A fleet watch matches when the fleet id is in the radio’s fleetIds. The first time that pair is seen, the alerter records a token of the watchlist id and the device key. The same radio does not beep again for that bookmark until it has left the live list and the token is dropped. A reappearance counts as new when the radio was first seen within the last 20 seconds, or when the current presence span started within the last 20 seconds. One pass beeps once, even if several radios qualified. Voice speaks one phrase, the class, the signature name, or both, after the beep. Vibration and the notification follow the flags on the watchlist row.

The default watchlist, the one a new install and Restore default signatures & presets write, is every built-in signature that has an Extra attention sentence, plus every built-in signature whose class is Drone. Extra attention itself is not the beep. It is the sentence. When the sentence is non-empty, a matching radio gets a mark on the live row, an amber card on the detail page, and a line in the sit report. A person can remove the bookmark and keep the sentence, or the other way around. The stock sentences for cameras say the hit is a pattern, and they say to look. That wording is the product’s, and the matcher does not add a confidence number on top of it.

15. The sit

A sit is a name and a start time, and then one row per radio heard while it is open. The word is Fieldwatch’s. It is not a Bluetooth or Wi-Fi term. Starting a sit copies the radios already on the live list, so a radio that was heard before the tap is in the report once the path has a point. The path itself starts empty. The first point is the next GPS fix after the start. The walk from before the tap is not in the line.

SitSession.ingest keeps a row only while the sit is open. The key is the same kind-and-address key as the live list. A second advertisement updates that row: the earliest first-seen, the latest last-seen, the louder of the signal extremes, the union of the signature ids, and a GPS trail of where this phone was. The trail is capped at 40 samples, spread across the time the radio was heard. An advertised aircraft position, heading, speed, and the operator position are kept beside that trail. The operator position is not drawn as the aircraft. The live decode chips are kept. The raw advertisement is not.

The cap is 6,000 radios. Past that, the session drops rows, oldest first, in two passes. The first pass drops Bluetooth rows that matched nothing and are not pinned. The second pass drops anything that is not pinned. A row is pinned when its key is a device-key bookmark, when one of its signatures is bookmarked, when it carries an advertised

position, or when a matched signature has an Extra attention sentence. A pinned row is not dropped to make room. If the table is still full of pinned rows, a new unpinned radio is refused.

The path is this phone's GPS, under the same hop limit as the rest of the application: a step that implies faster than about 150 kilometers an hour is not added. Points are kept about every 4 meters of travel, and a later point does not move the road already drawn. The cap on the path is 16,000 points. An open sit is written to the phone's private files about every 10 seconds. On the next launch the newest file still marked open is restored, so an update that restarts the application can resume it. Ten closed sits are kept. An eleventh drops the oldest.

The sit file also receives the other watches, as notes rather than as radios. A flood is a count and a time, not a place. A GNSS mark is a time, a level, and the text of what changed, with no coordinates. Those are stored so the report can put them on the path. They are not signature matches.

16. What the report decides

With no sit selected, Debrief and the path use the last 15 minutes. A selected sit uses that sit's window. Radios that belonged to a flood burst in the window are taken out of the inventory, so a burst of new addresses does not become a page of rows. The default report does not list every remaining radio. It leads with counts. A radio is listed when it has Extra attention, a custom name, a Mine mark, or a bookmark, or when Compare sees a decoded value that changed. Show all radios turns the inventory on. Show unmatched rotating BLE is a second switch, and with the short report on it adds nothing, because those addresses were never going to be listed. The counts include them either way.

The moving-with-you section is a test, not a second sensor. It runs on finder-class radios, on beacon-class radios, and on wearable-class radios, using the GPS trail on the sit row. The points are where this phone was when it heard the radio. The test refuses to judge when the phone's path has fewer than two points or spans less than 45 meters, or when the radio has fewer than two GPS samples. A radio whose trail spans less than 27 meters is called stationary even if the phone moved.

Own likely is the loud case. The loudest sample is at least -55 dBm, the quietest is at least -70 dBm, there are at least two samples, and the radio was still heard within the last three minutes. A trail that covers at least half of the phone's path length, is still that loud at the peak, and is still here, is the same verdict. A radio that was already present at the start, is still here, and was that loud, is own likely as well. Present at the start and still here, without that loudness, is moved with you. The sentences say a finder can be yours or can have been placed, a beacon does not follow anyone, and a wearable is typically a watch or a ring. A catalog sentence from a live decode chip is appended when the signature wrote one.

A possible tail is the remaining case: the radio was not present at the start, its trail spans at least 45 meters, and it has at least three samples. Three gates then push it back to stationary. If the last sample is 12 dB quieter than the peak, the phone walked away from a fixed radio. If fewer than two thirds of the samples are at least -75 dBm, it was a pass-by. If the trail's length is under half the phone's path, it was an arc, not a companion. What remains is called following. The sentence for a finder says to treat it as a possible tail until it is checked. None of these verdicts name a person.

Separated SmartTags are the one place the privacy id changes the report. Before the finder test, separated frames that share one privacy id are pooled into a single subject: the earliest address key, the earliest first-seen, the latest last-seen, the loudest and quietest signal, and the combined GPS samples. Those addresses then leave the test, so one tag is not judged three times because it rotated. A with-owner tag is not pooled. One separated advertisement, with no second address, remains a label. The pool is not a live row and is not saved. Mine keys are removed after

the verdict, so a radio marked mine does not appear in the following list.

Known cameras are not a result of this scan. The report can count a downloaded list of published places along the path, within about 150 meters, when the file is on the phone. A mark from that list was not heard on the air. The scan sections of the report are the radios. The camera count is a list someone else published.

Export writes the sit, not a new measurement. CSV and JSON lines are one row per radio, including the signature names, the Extra attention families, the custom name, the notes, and the mine flag. They do not include the path. A cell whose first character would be read as a formula by a spreadsheet is prefixed with an apostrophe. GPX, KML, and GeoJSON are the path plus one point per radio that has a GPS sample, at the loudest sample on that radio’s trail. The point is where this phone was. WiGLE is those points without the path. The ATAK data package is a zip of the same places with the live feed’s marker writer: the path, a hear-point, an advertised aircraft pin and its short trail, an operator pin only when that aircraft pin exists, and a GNSS hit as a pin on the nearest path sample. It does not redraw a flood, and it does not redraw the live GNSS ellipse. Privacy mode masks tails and coordinates on screen and in the report text. The log and the sit export keep the full values on purpose.

17. Parameters

Parameter	What the implementation uses
Signature record	Id, name, class, color, note, Extra attention, match-any, rules, peer fields, built-in, and an optional decode map. The class does not match.
Rule	Kind, text, company id, hex prefix, radio, and an enabled flag. No confidence number. Off stays in the record and is left out of the plan.
Stock file	fieldwatch-signatures, catalog 97. 255 signatures, 6,975 rules. 6,081 of the rules are OUIs.
Match	Any rule, except a match-all signature, which needs every rule. The stock file has one match-all row.
OUI index	Six hex characters. Separate maps for Wi-Fi addresses, Bluetooth addresses, and Wi-Fi vendor elements.
Protocol elements	00:50:F2 and 00:0F:AC are not a brand. They are left out of the vendor-element index.
Name glob	Anchored at both ends, case-insensitive. A question mark is any one character.
UUID	A four-character rule also matches the Bluetooth base UUID. A 128-bit UUID matches itself.
Live list	About 400 rows in the steady state, about 900 at the ceiling. An unnamed row ages out in about 3 minutes. A labeled row ages out in about 15.
Rematch	Skipped while the name, the UUIDs, the manufacturer bytes, and the vendor elements are unchanged. A peer rule forces a full pass.
Decode maps	19 in the stock file. 13 read manufacturer data. 6 read service data. 8 fields are marked for the live row. Gate first, then remainder, scale, and an added offset.
Bluetooth scan	One empty filter, every match, no batch delay. A fast session is recycled at about 70 seconds.
Wi-Fi scan	Four starts in two minutes, about 28 seconds apart. 8 seconds when Android throttling is off. A fresh result is one this application requested.
Alert	A bookmark. A new appearance is the first 20 seconds, or a presence that started in the last 20 seconds. One beep per pass.
Mine	That device key skips beep, voice, flash, vibration, and the notification. The signature’s other radios still alert.
Sit	One row per kind and address, cap 6,000. GPS trail 40 samples. Written about every 10 seconds. 10 closed sits.
Path	A point about every 4 meters. A hop faster than about 150 km/h is dropped. Cap 16,000 points.
Moving with you	Needs a phone path of at least 45 meters. Loud is -55 dBm. The on-body floor is -70 dBm. A 12 dB fade, or a trail under half the path, is not a tail.

References

1. Bluetooth Core Specification, Volume 6, Part B. Advertising events and the scan that receives them. Fieldwatch registers for the platform scan. It does not form an advertising packet of its own.

2. Android BluetoothLeScanner and ScanSettings. The callback, the scan mode, and the empty filter described above are this API. An empty filter is how the scan stays eligible while the screen is off on the phones this application has been run on.

3. Android WifiManager.startScan and the scan-results callback. A requested scan, rationed by the system. The results are what the application stores. It does not associate.

4. IEEE Registration Authority, organizationally unique identifiers. A registry prefix is not a signature. A prefix that belongs to a module vendor labels every product that uses the module.

5. Bluetooth SIG Assigned Numbers. Company identifiers and 16-bit member UUIDs. The stock rules cite these numbers. The assigned-number list itself is not imported.

6. ASTM F3411-22a, Standard Specification for Remote ID and Tracking. The 25-byte messages OpenDroneId parses. Operator location and aircraft location are different fields in that layout.

7. OpenDroneID, opendroneid-core-c. The heading byte and the message pack follow that reference code rather than a scale written in the signature file.

8. Celosia, G., and Cunche, M. “Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols.” *Proceedings on Privacy Enhancing Technologies*, 2020 (1), pp. 26–46. doi:10.2478/popets-2020-0003. Fieldwatch names the Continuity type. It does not recover the identifiers that paper showed how to recover.

9. Martin, J., Mayberry, T., Donahue, C., Foppe, L., Brown, L., Riggins, C., Rye, E. C., and Brown, D. “A Study of MAC Address Randomization in Mobile Devices and When it Fails.” *Proceedings on Privacy Enhancing Technologies*, 2017 (4), pp. 365–383. doi:10.1515/popets-2017-0054.

10. Becker, J. K., Li, D., and Starobinski, D. “Tracking Anonymized Bluetooth Devices.” *Proceedings on Privacy Enhancing Technologies*, 2019 (3), pp. 50–65. doi:10.2478/popets-2019-0036. The reason a signature prefers a name, a UUID, or a manufacturer prefix over the address.

A note on this paper

The scan, the signature file, and the sit described here are the ones in Fieldwatch 1.1.37, dated 29 October 2026, with stock catalog 97. A rule that is switched off stays in the signature and is left out of the plan. A radio that does not advertise is not in the list. A match is the pattern the file asked for. It is not the object, and it is not a person.