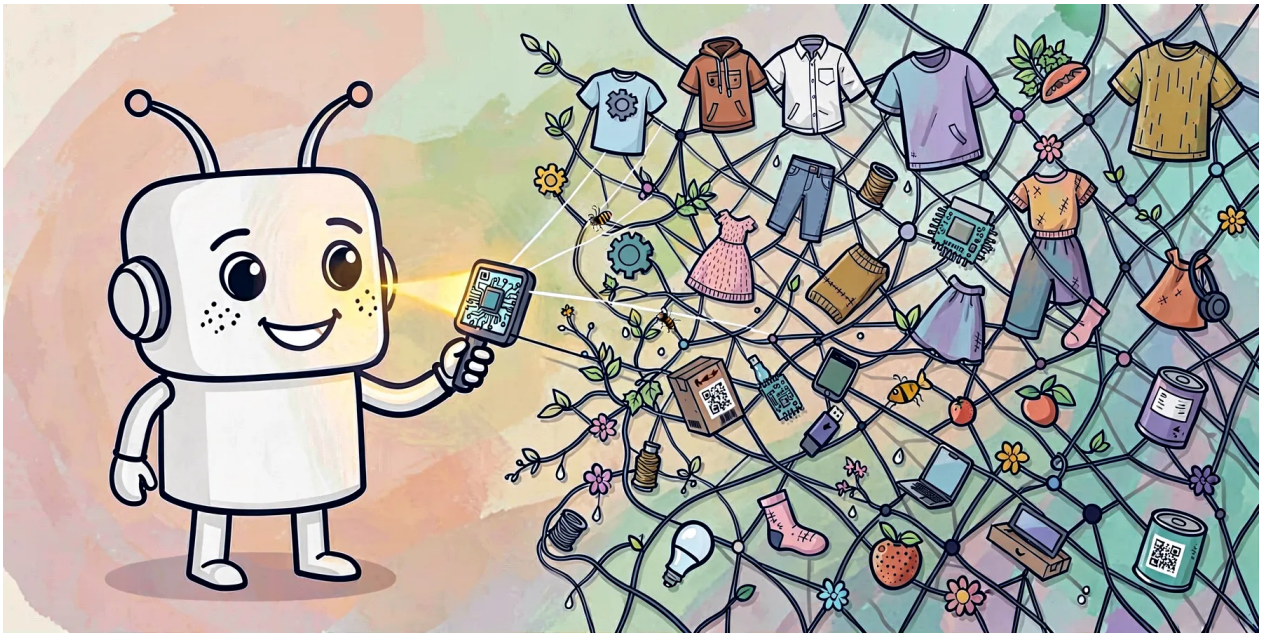
OpenEPCIS



The Web-based DPP doesn't need new infrastructure

Three GS1 standards — Digital Link, Web Vocabulary, Conformant Resolver — plus EPCIS 2.0 already cover the agent-facing layer of the Digital Product Passport.

Sven Böckelmann · 12 May 2026 · 19 min read

dpp • gs1 • epcis • digital-link • agentic-ai • linked-data • ontology • jsonld

openepcis.io/blog/dpp-stack-for-agents

Contents

The Web-based DPP doesn't need new infrastructure — three GS1 standards already cover the agent-facing layer

Walk into any DPP discussion in 2026 and you'll hear the same anxieties: regulatory deadlines, fragmented data, will-AI-eat-this. The conversations swirl around platforms, blockchains, walled gardens, "trust frameworks" — as if we were starting from zero.

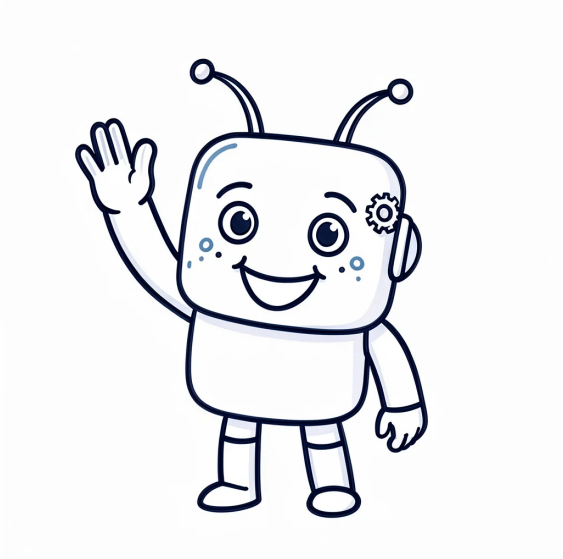
We're not.

CEN/CENELEC JTC 24 has ratified the **EN 18221 ff.** series for the Digital Product Passport. Eight standards covering identifiers, carriers, exchange protocols, persistence, APIs and interoperability. They are not aspirations. They are formal European norms, voted through, and ready to implement.

Three of them, in particular, form the stack that makes DPPs actually work for the customers who matter most: **machines**.

This article unpacks all three, and spends real time on the one piece of the stack that most platform pitches treat as an afterthought but is actually the load-bearing wall: **GS1 Digital Link**.

Why machines are now the customer



At GS1 Exchange in Cologne I made a claim that surprised a few people in the room (which has actually been obvious to me for more than a decade): the most important consumer of Digital Product Passport data won't be a human on a smartphone. It will be a machine, an agent.

By "agent" I mean the AI shopping assistants major retailers are quietly piloting. The agentic-commerce systems Salesforce, McKinsey and Deloitte have all published research on. The autonomous purchasing agents that look at a textile, a battery, a packaged food — and have to decide, in one shot, whether to buy it, route it, recall it, or trust it.

These agents do not read marketing copy. Show them this:

> "A wardrobe essential, reimagined. Crafted from NarraNatureTex™, our signature heritage fibre for a soft, lived-in hand-feel…"

…and ask "what is the material?" An LLM will guess. A purchasing agent that guesses costs money.

Show them this:

```
"gs1:textileMaterialDescription": {
  "en": "Cotton",
  "de": "Baumwolle",
  "fr": "Coton",
  "es": "Algodón",
  "nl": "Katoen"
}
```

…and the answer is unambiguous, multilingual, type-safe, queryable. The agent doesn't guess. It knows.

That's not a UX detail. It's the difference between data an agent can act on and data it has to guess at — and it scales from one product to a billion, in the same shape.

The breakthrough you've already been using: GS1 Digital Link

Before we get to the three standards, we have to talk about the URL syntax that makes the rest of the stack possible.

For thirty-plus years GS1 identifiers lived inside barcodes and EDI messages. The `(01)` GTIN, the `(21)` serial, the `(10)` batch, the `(17)` expiry — Application Identifiers, all parsable, all unambiguous, all alien to the web. To put them on the web you needed an EPC URN:

```
urn:epc:id:sgtin:4068977.934335.19odefMoeoIbBwfuFZa5
```

Technically correct. Practically opaque. Not dereferenceable. Not crawlable. Not a URL.

GS1 Digital Link does something almost embarrassingly obvious in retrospect: it expresses the same identifier as a *real* URL.

```
https://id.example.com/01/04068977934335/21/19odefMoeoIbBwfuFZa5
```

Read it once and the structure tells you everything:

- `https://id.example.com` — a host, resolvable by every browser, every cURL, every fetch in every language since 1995
- `/01/04068977934335` — Application Identifier `01` is GTIN; the value follows
- `/21/19odefMoeoIbBwfuFZa5` — Application Identifier `21` is the serial

That's it. **Path segments alternate AI and value.** Any GS1 identifier — and there are dozens — slots into the same shape:

Identifier	Application Identifier	Digital Link path
GTIN (trade item)	`(01)`	`/01/04068977934335`
GTIN + serial (SGTIN)	`(01)(21)`	`/01/04068977934335/21/SN-001`
GTIN + batch	`(01)(10)`	`/01/04068977934335/10/LOT-A42`
SSCC (logistic unit)	`(00)`	`/00/304068977000000017`
GLN (location)	`(414)`	`/414/4068977000000`
GLN + extension	`(414)(254)`	`/414/4068977000000/254/dock-7`
GRAI (returnable asset)	`(8003)`	`/8003/04068977000000A42`
GIAI (individual asset)	`(8004)`	`/8004/4068977-FORK-17`
GDTI (document)	`(253)`	`/253/4068977000000DOC42`

Non-ID key qualifiers — expiry, weight, best-before, country of origin — sit in the **query string**, where qualifiers belong on the web:

```
https://id.example.com/01/04068977934335/21/SN-001?17=261231&3103=001750
```


`(17)=261231` is "expires 2026-12-31", `(3103)=001750` is "1.750 kg". Same Application Identifiers as the barcode. Same values. Just expressed in the syntax the web already speaks.

Why this matters more than it looks

Three things fall out of this design that no proprietary URL shortener can replicate:

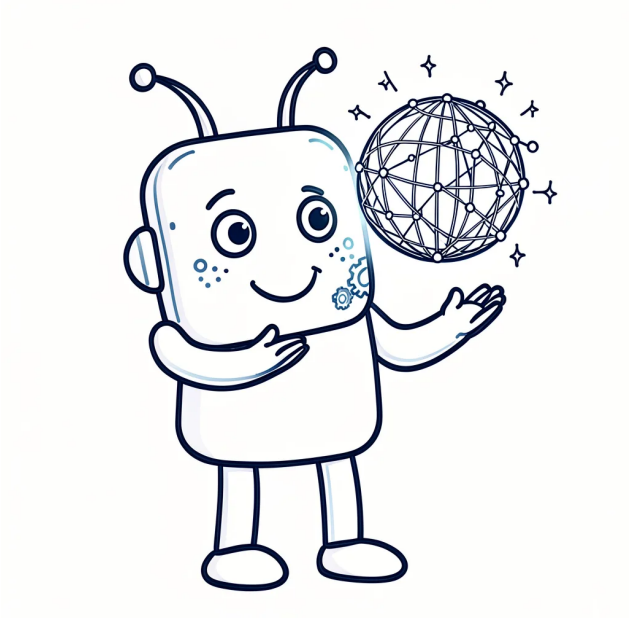
1. Bidirectional mapping is mechanical. A Digital Link URL maps to an EPC URN and back without semantic loss. The two are isomorphic — same identifier, two encodings, both standardised. Legacy EPCIS systems speak URN; web-native systems speak Digital Link; the bridge is a regex, not a vendor.

2. The same string is the QR target *and* the API endpoint. A camera reads the QR, the browser opens the URL, the agent does an HTTP GET on the URL — and it's all the same string. No "look up the GTIN, then look up where its data lives, then…". The identifier *is* the locator.

3. Every HTTP tool ever written works on it. `curl`, `wget`, `fetch()`, `requests.get()`, an Apache log line, a CDN cache key, an OpenAPI path, a load balancer rule. None of them needed a single line of GS1-aware code to handle a Digital Link URL. That is not a small thing — it is the entire reason the standard scales.

It is, in the original sense of the word, **genius and obvious**: take an identifier scheme that already used hierarchical, prefix-stable, AI-tagged segments, and notice that this is *exactly* what URL paths are for. The barcode had been a URL all along. GS1 Digital Link just admits it.

This single design choice is what makes the next three standards work as a stack instead of as three disconnected specs.



Standard 1 — GS1 Web Vocabulary: the language agents can actually read

EN 18223 covers system interoperability, and at its heart sits the **GS1 Web Vocabulary**: an ontology of GS1 concepts published as JSON-LD, aligned with schema.org, validatable with SHACL, queryable with SPARQL.

This is the part most platform pitches gloss over. Vocabulary isn't just labels — it's a knowledge graph. `gs1:Clothing`, `gs1:textileMaterialDescription`, `gs1:countryOfOrigin`, `gs1:sizeCodes` — every one is a node with defined semantics, regulatory anchoring, and a clear path to Linked Data.

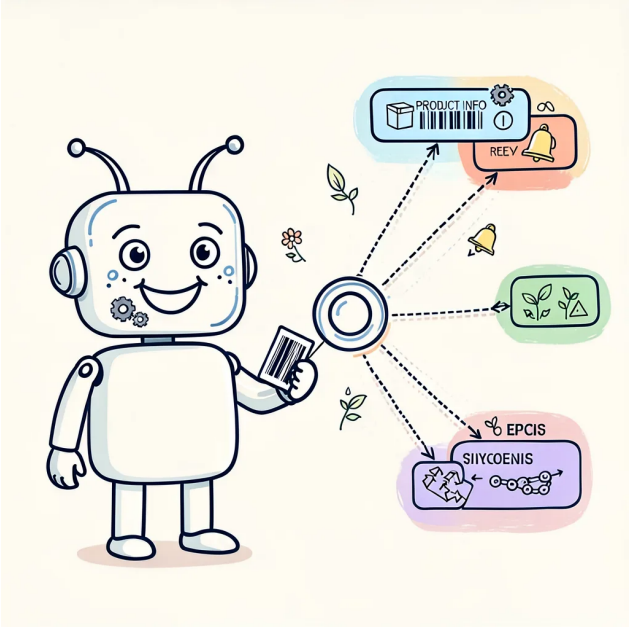
A short example of what an agent actually receives:

```
{
  "@context": "https://ref.openepcis.io/extensions/eu/textile/context.jsonld",
  "id": "https://id.example.com/01/04068977934335",
  "type": "gs1:Clothing",
  "gs1:textileMaterialDescription": {
    "en": "Cotton",
    "de": "Baumwolle"
  },
  "gs1:countryOfOrigin": "PT",
  "eudr:dueDiligenceStatement": {
    "id": "https://eudr.example.com/dds/REF-2026-00042",
    "type": "eudr:DueDiligenceStatement"
  }
}
```

When an AI agent sees this, it doesn't have to translate marketing prose into intent. The semantics *are* the intent — which unlocks the downstream behaviour everyone wants from a DPP: authenticity checks, recall handling, sustainability scoring, customs decisions, automated buying.

Without an ontology, every brand reinvents its own data model and every agent has to relearn. With GS1 Web Vocabulary, the agent learns the language once — and reads every brand on the planet that conforms.

We've been building on top of it at OpenEPCIS, and you can browse the result live at ref.openepcis.io. More on that below.



Standard 2 — GS1 Conformant Resolver: identity that resolves to endpoints

EN 18216 covers data exchange, and the canonical answer is the **GS1 Conformant Resolver** paired with GS1 Digital Link.

Take the Digital Link URL from earlier:

`https://id.example.com/01/04068977934335/21/19odefMoeoIbBwfuFZa5`

That URL is more than a QR target. It is a **federated discovery mechanism**. Hit it with an `Accept:`

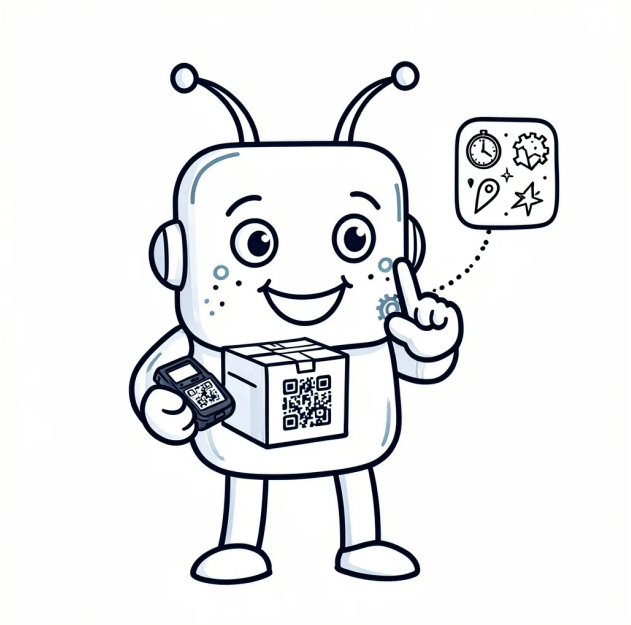
`application/linkset+json` header and you get back a structured **link set** — a list of typed endpoints the agent can interpret without any prior agreement with the brand:

```
{
  "linkset": [{
    "anchor": "https://id.example.com/01/04068977934335/21/19odefMoeoIbBwfuFZa5",
    "https://gs1.org/voc/pip": [
      { "href": "https://example.com/products/sweater-001", "type": "text/html", "hreflang": ["e
    ],
    "https://gs1.org/voc/recyclingInfo": [
      { "href": "https://example.com/recycle/textile/cotton", "type": "text/html" }
    ],
    "https://gs1.org/voc/epcis": [
      { "href": "https://epcis.example.com/events?eq.epc=...", "type": "application/ld+json" }
    ],
    "https://eudr.example.com/voc/dueDiligence": [
      { "href": "https://eudr.example.com/dds/REF-2026-00042", "type": "application/ld+json" }
    ]
  }]
}
```

Each link relation tells the agent *what kind of resource* it is pointing at. Product information page. Recycling guidance. EPCIS event history. Compliance certificate. Standardised across every brand's resolver — that's what *conformant* means.

This is what traditional URL shorteners and walled gardens cannot replicate. An agent that learned to talk to one resolver can talk to all of them. A barcode that talks to a resolver becomes the front door of a DPP — no app, no platform lock-in, no proprietary SDK.

And — quietly — the same URL serves humans too. A browser following it without the JSON Accept header gets redirected to the right localised product page. One identifier. One URL. Two audiences. Same standard.



Standard 3 — EPCIS 2.0: the verified history

EN 18221 and EN 18222 cover storage, persistence and APIs, and **EPCIS 2.0** is the implementation that makes them concrete.

EPCIS answers the five questions every supply-chain participant — and every AI agent — has to answer when they hand something on:

What. When. Where. Why. How.

A simplified event:

```
{
  "@context": ["https://ref.gs1.org/standards/epcis/epcis-context.jsonld"],
  "type": "ObjectEvent",
  "eventTime": "2026-04-29T08:14:00Z",
  "eventTimeZoneOffset": "+02:00",
  "epcList": [
    "https://id.example.com/01/04068977934335/21/19odefMoeoIbBwfuFZa5"
  ],
  "action": "OBSERVE",
  "bizStep": "shipping",
  "disposition": "in_transit",
  "readPoint":   { "id": "https://id.example.com/414/4068977000000" },
  "bizLocation": { "id": "https://id.example.com/414/4068977000000" }
}
```

Notice what every identifier in there has in common: they're all **Digital Link URLs**. The same syntax that drove the barcode and the resolver now identifies the EPC, the read point, and the business location inside the event. One identifier scheme, top to bottom. Drop the EPC URL into a browser and you arrive at the same product. Drop it into the resolver and you get the link set. Drop it into EPCIS and you get the history. Same string, three different reads.

Each event is a JSON-LD document tied to a GTIN or serialised identity, time-stamped, geo-located, business-step labelled, and — as EN 18246 lands — cryptographically signed via W3C Verifiable Credentials.

Worth saying loudly: **this is not a forecast.** EPCIS events have been driving supply-chain decisions for more than a decade — best-before predictions, FEFO routing, spoilage prevention, real money saved. The events are real, the repositories are real, the standard works.

What is new is the consumer of those events. Yesterday it was a temperature-monitoring service. Today it is an agentic purchasing system asking "is this batch authentic, compliant, fresh, ethically sourced — yes or no?"

Same events. New audience. Same answer: a verified, queryable, JSON-LD repository of what actually happened to this product.

How the three standards lock together

Treated separately, these standards each look like a niche concern. Treated together — and bound by Digital Link — they describe a single, coherent architecture: the *Web DPP Stack*.

1. **GS1 Digital Link** — the *identity layer*: a web-native URL that **is** the identifier
2. **GS1 Web Vocabulary** — the *semantic layer*: what the data means
3. **GS1 Conformant Resolver** — the *discovery layer*: where the data lives
4. **EPCIS 2.0** — the *temporal layer*: the verified history of what happened

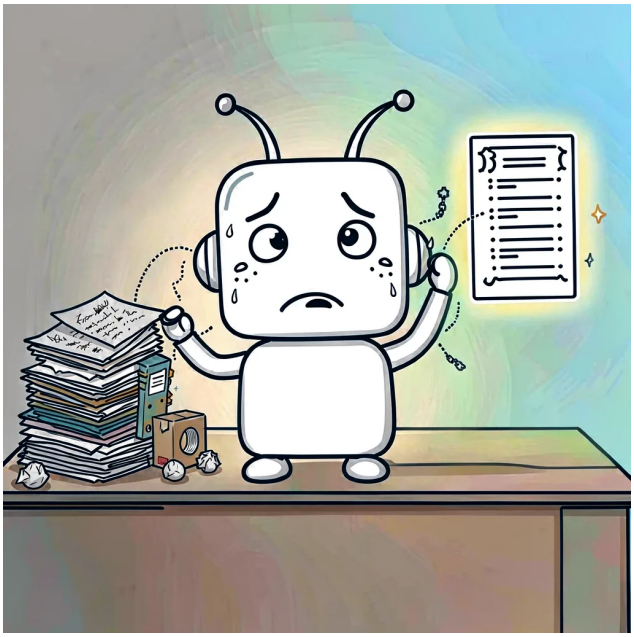
Each layer needs the others. Without Digital Link, there is no shared addressing scheme — every layer would invent its own and the stack would be three boxes connected by glue code. Without the Vocabulary, the resolver returns opaque URLs. Without the resolver, the vocabulary has no distribution mechanism. Without EPCIS, you get static snapshots with no provenance.

Put them together and you get a Digital Product Passport machines can read, trust, and act on — at scale.

> **A note on EDI and GDSN.** None of this replaces what already works. EDI and GDSN do the heavy lifting between trading partners today and they are not going anywhere. But both assume bilateral onboarding and signed agreements — they were never meant to be frictionless. The standards above add **W3C-standards-based, role-based access for public and regulatory consumers** on top of that foundation, for the strangers who increasingly need to read your product data: agents, customs, recall systems, regulators.

The real work: mapping legacy data into the regulated space

Here is the part nobody puts on a conference slide. Your product data does not currently live in a JSON-LD graph aligned to GS1 Web Vocabulary. It lives in an ERP, in a PIM, in an EDI feed, in the GDSN data pool, or in a spreadsheet a procurement team has been editing since the beginning of time — maybe even still on paper. Neither the regulator nor the agent cares about any of that; they want the answer in the language of the regulation.



That gap — between the legacy schema you have and the regulated, semantic, machine-readable schema you *need* — is the actual DPP project, and nobody who ships a product escapes it.

It is also where **JSON-LD and the semantic web stack earn their keep**.

JSON-LD lets the *same* document carry the field your developer wrote and the meaning the regulator demands. SHACL validates it. SPARQL queries it. `owl:equivalentClass` and `skos:exactMatch` declare that *your* `material_code = "CO"` *is* `gs1:cottonFibre`, without rewriting your source systems. The graph is the contract, not yet another bespoke integration.

Don't mistake "no rewriting" for "no work". Mapping a real product portfolio onto a regulated ontology is its own engineering programme. It involves:

- **Regulatory interpretation** — which classes and properties apply to *your* product, under which directive, with which evidence requirements (textile composition declarations, battery passport carbon footprint, EUDR plot geolocation, FSMA 204 critical tracking events).
- **Schema reconciliation** — your internal codes mapped against GS1 Web Vocabulary, the EU SEMIC Core Vocabularies and schema.org. Most of the work is deciding what *isn't* a 1:1 match and what to do about it.

- **Master-data hygiene** — GTIN allocation policy, GLN cleanup, serialisation and batch numbering, party identifiers. The mapping is only as honest as the underlying master data.
- **Capture infrastructure** — where do EPCIS events actually come from? ERP, MES, WMS, scanners, IoT? Every emitter is its own integration project.
- **Resolver hosting and link-set governance** — who owns the link relations, who updates them, who's accountable when a recall URL goes stale.
- **Label, packaging and physical output** — Digital Link URLs only matter once the QR or RFID actually carries them on the shelf.
- **Change management** — regulations move, vocabularies evolve, your portfolio changes. The mapping is a living artefact, not a one-off.

Doing this *once* on a semantic foundation is the win — every downstream consumer (agent, customs officer, recall system, regulator) reads the same canonical truth, instead of you maintaining N bespoke integrations forever. Doing it on top of a proprietary platform locks you in, locks the regulator out, and locks the agent out too.

This is the technology choice that makes the rest of the stack possible. Without a semantic web foundation, every connector is a one-off; with it, every connector is a contribution to a shared graph.

This is the work we focus on at OpenEPCIS: building the mapping layer so a single JSON-LD document can answer the regulator, the customs system and the customer's agent without three separate integrations.

OpenEPCIS DPP-Ready: the mapping layer in code

OpenEPCIS DPP-Ready is our Apache-2.0 contribution to the mapping work above. Not a platform, not a product — but the result of a multi-year, full-time effort by our team to read every relevant regulation, ratified European norm and current draft, reconcile their concepts against GS1 Web Vocabulary, the EU SEMIC Core Vocabularies and schema.org, and ship the alignment as code that other organisations can pull in instead of repeating the same exercise in isolation. What you get:

- An **ontology library** of currently 120+ classes and 430+ properties, published as TTL and JSON-LD
- **Cross-vocabulary alignment** between GS1, the EU SEMIC Core Vocabularies and schema.org, expressed in `owl:equivalentClass` and `skos:exactMatch`
- A **validator** covering eight regulations today: Battery, EUDR, Textile, Electronics, CPR, PPWR, Detergents and FSMA 204
- A live browseable ontology at **ref.openepcis.io**

A few design choices worth flagging:

What it does	Why we built it that way
Dereferenceable ontologies at `https://ref.openepcis.io/extensions/...`	Every class and property gets a stable, HTTP-resolvable URI
JSON-LD contexts shipped alongside the ontology	No bespoke context wiring per integration — point at `@context` and go
EPCIS 2.0 extension architecture	Regulation-specific behaviour activated via the `GS1-Extensions` header
Cross-standard alignment	Anchors out to UNTP, CIRPASS-2 and CEN/CENELEC JTC 24 instead of forking

The guiding rule: walk down the stack and use the *most authoritative* vocabulary that already covers your concept. Within Layer 1 we check **GS1**, then **SEMIC**, then **schema.org**, and only mint a new IRI when none of the three has it. Less novelty is the goal, not more.

A tour of ref.openepcis.io

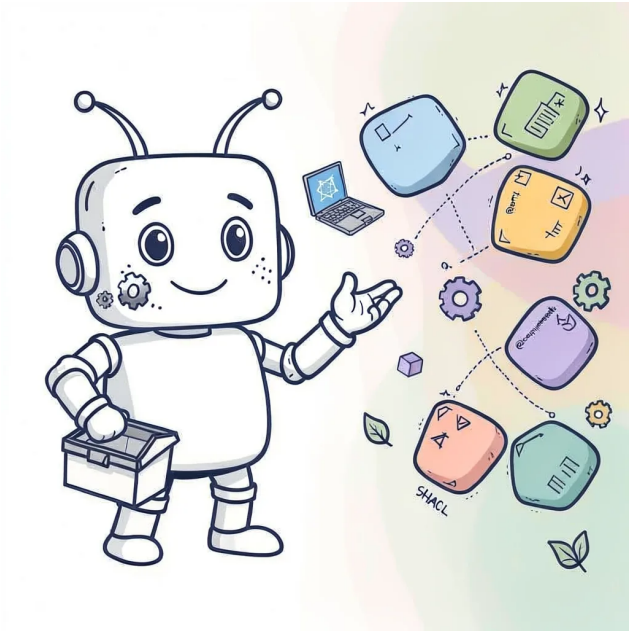
We've just made **ref.openepcis.io** publicly browseable — a dereferenceable vocabulary site for the DPP ontologies we maintain.

The structure mirrors the regulation landscape:

URL	What lives there
ref.openepcis.io/extensions/common/core/	Cross-cutting classes and properties shared across regulations — units, parties, references, certificates
ref.openepcis.io/extensions/eu/battery/	EU Battery Regulation: chemistry, capacity, recycled content, carbon footprint
ref.openepcis.io/extensions/eu/textile/	EU Textile Regulation: fibre composition, country of origin, care, microfibre release
ref.openepcis.io/extensions/eu/eudr/	EU Deforestation Regulation: due-diligence statements, geolocation of plots, commodity types
ref.openepcis.io/extensions/eu/electronics/	Energy efficiency, repair index, spare-part availability
ref.openepcis.io/extensions/eu/ppwr/	Packaging and Packaging Waste Regulation
ref.openepcis.io/extensions/eu/cpr/	Construction Products Regulation
ref.openepcis.io/extensions/eu/detergents/	Detergents Regulation
ref.openepcis.io/extensions/us/fsma204/	FSMA Section 204 (US food traceability)

Every page is generated from the TTL in the GitHub repository. The TTL is canonical; the website is just a view onto it. Each class and property has its own resolvable URI you can copy-paste into a JSON-LD `@context`, validate against with SHACL, query with SPARQL, or hand to an LLM that already knows how to read RDF.

The point is mundane and important: a standard you can resolve in a browser, copy-paste a JSON-LD context from, and validate against — instead of a PDF that ages on a shared drive.



What this means for your DPP roadmap

Much of the DPP conversation is still framed as a platform choice. It mostly isn't one. The standards are documented, ratified, and there is working open-source code — ours and others' — that implements them today.

Eight ratified European standards. Three of them load-bearing for the agentic era, sitting on top of the EDI and GDSN rails the industry already trusts. None of them research projects waiting to be invented. And all of them held together by a single design decision — that a GS1 identifier is just a URL — that the rest of the web has been quietly waiting for since 1995.

GS1 Digital Link + GS1 Web Vocabulary + GS1 Conformant Resolver + EPCIS 2.0 — the identity, semantic, discovery and temporal layers — are what turn a regulated data file into something a machine can actually consume.

If your products carry a GTIN, you are already in this ecosystem. The DPP question is no longer whether to use these standards — it is how to map your existing data into them. The consumer at the other end is increasingly an agent, and an agent needs the data in a language it can read.

A practical sequence:

1. **Pick one regulation that already applies to one of your products** — Textile, Battery, EUDR, FSMA 204, whichever bites first. Scope it to a single product family, not the whole portfolio.
2. **Map that product family onto the corresponding ontology at ref.openepcis.io.** Field-by-field. Expect this to surface gaps in your master data — that *is* the work, not an obstacle to it.
3. **Issue Digital Link URLs for those products.** `/01/<gtin>/21/<serial>` is enough on paper; getting your serialisation, label printing, packaging line and ERP to actually emit those URLs is a real, cross-functional project that touches operations, not just IT.
4. **Stand up a resolver** that returns a link set per identifier. A static file per product is a valid first version, but the link relations and the documents they point at — product information page, recycling guidance, recall procedure, due diligence statement — are content that has to be authored, owned and kept current.
5. **Wire EPCIS event emission into the lifecycle moments that matter** — manufacture, ship, receive, sell, recall. This is capture infrastructure: ERP, MES, WMS, scanners, IoT — each one its own integration.

None of these steps requires a platform purchase or a research project, and that is the genuinely good news. But none of them is a weekend's work either. Steps 2, 3 and 5 are typically months of engineering and operational change for any organisation past prototype scale, and that's *with* OpenEPCIS DPP-Ready and the GS1 standards doing the heavy intellectual lifting underneath. The value of having ratified standards and a public ontology is not that the work disappears — it's that you do it *once*, on shared foundations, and every regulator, customs system and AI agent on the receiving end reads the same canonical answer.

That is the realistic shape of a DPP programme: bounded engineering on shared standards, instead of unbounded platform R&D.

Sven Böckelmann is Tech Lead at benelog and a contributor to OpenEPCIS — an Apache-2.0 implementation of GS1 Web and Visibility standards. Browse the live DPP ontology at ref.openepcis.io. The companion short-form version of this article is on LinkedIn.

A note on the images: yes, they're AI-generated too — first sketched with Gemini, then refined locally in Draw Things with FLUX.2 [klein]. Hope they weren't too distracting — felt fitting for an article about agents reading product data. ;-)

#DigitalProductPassport #GS1 #EPCIS #DigitalLink #AgenticAI #LinkedData #SupplyChain #Ontology #JSONLD