

BDF Standards Stack

Implementation Reference for the Bio-Digital Footprint Protocol

Fair Food Data (FFD)

Version 1.0 · 24 September 2026

Abstract

This paper documents the open standards stack on which the Bio-Digital Footprint (BDF) protocol [Fair Food Data, 2026a] is implemented. The protocol composes nine open standards spanning canonical encoding, content addressed storage, sensor telemetry, supply chain events, food domain ontology, agricultural sensor ontology, provenance graphs, verifiable credentials, and public chain anchoring. None is invented for this protocol; each is selected for its maturity, governance, and adoption profile in the relevant industrial domain.

The paper provides the canonical mapping from each BDF attestation vector and primitive to its standards based payload, with worked examples in JSON-LD and beeson canonical form. The intended audience is engineers building BDF compliant deployments, who require explicit field by field mapping rather than a high level architecture description, and standards body assessors evaluating the protocol's interoperability claims.

The composition principle is to layer the standards rather than reinvent: BDF is a coordinator of existing standards, not a competitor. The selection follows three criteria: (i) open governance under a recognised body (W3C, GS1, IETF, ISO, OBO Foundry), (ii) production maturity with deployments at industrial scale, and (iii) extensibility to new domains without breaking existing schemas. The result is a stack that an integrator can adopt incrementally and that registry assessors can verify against published standards documents.

A complete worked example walks a single sensor reading through every layer of the stack, from the field IoT device's SenML payload through GS1 EPCIS event wrapping, FoodOn cultivar ontology, PROV-O provenance link, W3C Verifiable Credential issuance, beeson canonical serialisation, BMT hashing, and Swarm Single Owner Chunk anchoring. The example is grounded in real pilot data from the BDF main paper [Fair Food Data, 2026a] and shows end to end traceability from physical observation to cryptographic commitment.

Standards versioning, deprecation handling, and migration strategy are documented in the closing section, supporting deployments that must remain compliant as the underlying standards evolve.



Contents

1	Introduction and Scope	5
1.1	Purpose	5
1.2	Design Principle: Composition over Reinvention	6
1.3	Relation to the Other BDF Papers	6
1.4	Scope Boundaries	7
1.5	Notation	7
2	Stack Overview	8
2.1	Layered Architecture	8
2.2	Selection Criteria	9
2.3	Standards Selection Rationale	10
2.4	Standards Composition Rules	10
3	Canonical Encoding: beeson	11
3.1	Beeson Type System	11
3.2	Canonical Serialisation Rules	12
3.3	Binary Merkle Tree Hashing	12
3.4	Schema Versioning	12
3.5	Composition with Higher Layers	13
3.6	Reference Implementation	13
4	Storage and Revocation: Swarm	14
4.1	Single Owner Chunks	14
4.2	Postage Stamps and Retention	14
4.3	Encryption Envelope	15
4.4	DREAM References	15
4.5	Mapping BDF Primitives to Swarm	16
5	Sensor Telemetry: SenML	16
5.1	SenML Record Structure	17
5.2	Units and Measurement Types	18
5.3	Time Anchoring	18
5.4	Aggregation at the Gateway	18
6	Supply Chain Events: GS1 EPCIS 2.0	19
6.1	EPCIS Event Types	19
6.2	Core Business Vocabulary	20
6.3	Identifiers: GLN, GTIN, EPC	20
6.4	EPCIS to BDT Bridge	21
6.5	FSMA Section 204 Compatibility	22
7	Food Domain Ontology: FoodOn	22
7.1	FoodOn Class Hierarchy	22
7.2	Mapping BDF Vectors to FoodOn	23

7.3	Tier 3 Classifier and FoodOn	23
7.4	Process Descriptor Mapping	23
7.5	Provenance of the FoodOn IRI	24
8	Agricultural Device Ontology: SAREF4AGRI	24
8.1	SAREF Core Concepts	24
8.2	SAREF4AGRI Device Classes	25
8.3	Device Identity in BDF	25
8.4	SAREF Measurements and BDF Vectors	26
8.5	Multi Vertical Extensibility	26
9	Provenance Graph: PROV-O	27
9.1	PROV-O Core Classes	27
9.2	Mapping BDT Lifecycle to PROV-O	27
9.3	Worked Example: Transformation Provenance	27
9.4	Distribution Provenance	29
9.5	Verifier Walks the Graph	30
9.6	Composition with Other Layers	30
10	Credential Envelope: W3C Verifiable Credentials 2.0	30
10.1	VC 2.0 Structure	30
10.2	BDF Vector as Verifiable Credential	31
10.3	Decentralized Identifiers (DIDs)	32
10.4	Cryptographic Suites	32
10.5	Verifier Logic	33
10.6	Selective Disclosure with Verifiable Credentials	33
10.7	Status and Revocation	33
11	Public Chain Anchoring	34
11.1	What Gets Anchored	34
11.2	Reference Chains	34
11.3	Smart Contract Surface	35
11.4	Settlement vs Anchoring	35
11.5	Gas Cost and Throughput	36
11.6	Chain Migration and Multi Chain Deployments	36
12	Vector Schema Mappings	36
12.1	Agronomic Vector $\vec{A}$	37
12.2	Logistic Vector $\vec{L}$	37
12.3	Process Vector $\vec{P}$	37
12.4	Recipe Vector $\vec{R}$	37
12.5	External Attestation $\vec{E}$	37
13	Worked Example: One Reading Through the Stack	38
13.1	Layer 0: Physical Reality	40
13.2	Layer 1: SenML Encoding	40

13.3	Layer 2: Domain Semantics (FoodOn + SAREF4AGRI)	40
13.4	Layer 3: PROV-O Provenance Graph	41
13.5	Layer 4: W3C Verifiable Credential Envelope	41
13.6	Layer 5: Beeson Canonical Encoding	42
13.7	Layer 6: Swarm Storage	42
13.8	Layer 7: Public Chain Anchoring	42
13.9	End to End Verification	43
13.10	Summary of Layers Crossed	43
14	Versioning and Migration	43
14.1	Standards Version Pinning	44
14.2	Compatible vs Breaking Revisions	44
14.3	Cryptographic Primitive Migration	44
14.4	FoodOn Class Evolution	45
14.5	EPCIS and CBV Vocabulary Updates	45
14.6	Migration Procedure	45
14.7	Deprecation of the BDF Stack Itself	46
15	Conclusion	46
	Document History	48

1 Introduction and Scope

The Bio-Digital Footprint (BDF) protocol [Fair Food Data, 2026a] delivers verifiable records for transformed bioassets through a typed digital twin and a tiered attestation model. The main paper describes the protocol structurally; the methodology paper [Fair Food Data, 2026b] applies it to recovered food carbon credits; the security analysis paper [Fair Food Data, 2026c] formalises its threat model. This paper provides the implementation reference: the canonical mapping from each BDF primitive to the open standards on which it is built.

1.1 Purpose

The protocol is designed to be implemented, not merely described. An engineer building a BDF compliant deployment needs:

- Field by field correspondence between the abstract attestation vectors of the main paper and concrete schemas in published standards.
- JSON, JSON-LD, and beeson canonical examples that compile and verify against publicly available reference implementations.
- Versioning policy that survives standards evolution without forcing protocol revisions.
- Worked examples that walk a single observation through every layer of the stack.

This paper provides each. Where the protocol composes an existing standard, we cite the standard and

document the composition rule. Where the protocol extends a standard with a vector specific schema, we state the extension formally in beeson and provide the canonical mapping to the standard’s data model.

1.2 Design Principle: Composition over Reinvention

The principal design choice is to compose existing standards rather than invent new ones. Concretely, the protocol adopts:

- **beeson** [Tron et al., 2026] for canonical typed object encoding and BMT hashing.
- **Swarm** [Trón, 2024] for content addressed storage with Single Owner Chunks, postage stamps, and DREAM revocation.
- **SenML** [Jennings et al., 2018] for sensor measurement payloads.
- **GS1 EPCIS 2.0** [GS1, 2022b] for supply chain event semantics.
- **FoodOn** [Dooley et al., 2018] for the food domain ontology.
- **SAREF4AGRI** [ETSI, 2020b] for the agricultural sensor and device ontology.
- **PROV-O** [W3C, 2013] for the provenance graph.
- **W3C Verifiable Credentials 2.0** [W3C, 2024a] for the credential and signature envelope.
- **Ethereum and Gnosis Chain** [Wood, 2014, Gnosis, 2024] for public chain anchoring.

Each standard is selected against three criteria: open governance under a recognised body (W3C, GS1, IETF, ISO, OBO Foundry), production maturity with deployments at industrial scale, and extensibility to new domains without breaking existing schemas. Section 2 elaborates the selection rationale for each.

1.3 Relation to the Other BDF Papers

The BDF programme comprises four documents:

[Fair Food Data, 2026a] Main paper.

The structural definition of the protocol: Bio-Digital Twin, attestation vectors, tiered attestation, three layer architecture, Hardening Pathways. It is the conceptual reference and the entry point for new readers.

[Fair Food Data, 2026b] Methodology.

The application of the protocol to recovered food carbon credits, including counterfactual derivation, baseline emission factors, project emissions accounting, uncertainty assessment, and cross-walk to ICR, Verra, and Gold Standard. It is the deployment reference for carbon market projects.

[Fair Food Data, 2026c] Security Analysis.

The formal threat model, trust assumptions, and per tier security reductions. It is the assessment reference for ICVCM CCP labelling and peer review.

Standards Stack (this paper).

The implementation reference: canonical mapping from each protocol primitive to its standards based payload, with worked examples and versioning policy.

The four papers are independent in scope but cross referencing. A reader interested in implementation

should consult this paper alongside the main paper for structural context. A registry assessor should consult this paper alongside the methodology paper for compliance verification.

1.4 Scope Boundaries

In scope.

- The mapping from each BDF primitive (BDT, attestation vectors, distribution events, External Attestations) to standards based payloads.
- Canonical encoding rules and conformance requirements for each standard.
- Worked examples in JSON, JSON-LD, and beeson canonical form.
- Versioning and migration policy across standards revisions.

Out of scope.

- Performance benchmarks of reference implementations. These are deployment specific and depend on hardware, network, and cohort sizes.
- Specific reference implementations in any programming language. The standards are language agnostic; reference implementations are released as separate engineering artefacts.
- Complete API specifications for the BDF protocol’s verifier and detector interfaces. These are documented in the protocol’s reference implementation repository.
- Detailed security analysis of the standards themselves. We rely on the standards bodies’ own assurance processes; security reductions of the BDF protocol are in [Fair Food Data, 2026c].

1.5 Notation

This paper uses the following conventions:

Monospace

For literal field names, identifiers, and code.

italic

For type names and abstract concepts.

bold

For canonical standard names on first introduction.

@

Prefix for JSON-LD context terms (e.g. @id, @type).

ex:

Namespace prefix for example terms in JSON-LD.

prov:

Namespace prefix for PROV-O terms.

foodon:

Namespace prefix for FoodOn IRIs.

saref4agri:

Namespace prefix for SAREF4AGRI terms.

gs1:cbv:

Namespace prefix for GS1 Core Business Vocabulary.

Worked examples are presented in the most appropriate format for the layer being illustrated: SenML in JSON, EPCIS in JSON-LD, FoodOn in OWL turtle (in summary form), beeson in its native bracket notation, and BMT hashes as 64 character hexadecimal strings prefixed with 0x.

2 Stack Overview

The BDF stack is layered. Each layer addresses a distinct concern and is implemented by an open standard. This section presents the layered diagram, identifies the standard at each layer, and rationalises the selection.

2.1 Layered Architecture

The stack comprises seven layers from the physical sensor at the bottom to the public chain anchor at the top. Figure 1 shows the composition.

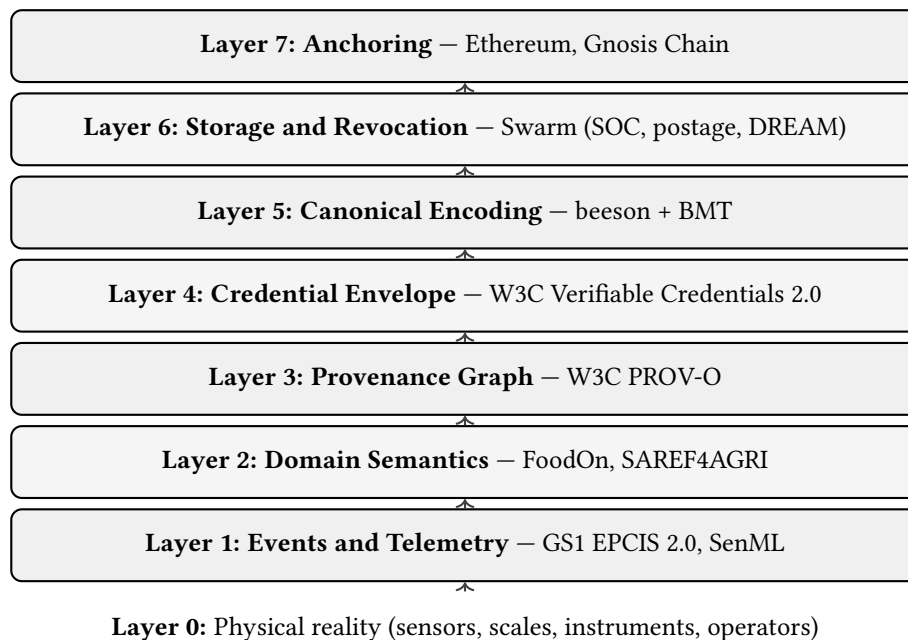


Figure 1: The BDF standards stack. Each layer is implemented by one or more open standards. Data flows upward from physical reality to public chain anchor; verification flows downward.

Reading the layers from the bottom.

1. **Layer 0: Physical reality.** Soil sensors, weather stations, scales, GPS gateways, SCADA systems, laboratory instruments. Outside the protocol’s scope; the protocol assumes Layer 0 is calibrated within its declared uncertainty (the Tier 2 statistical guarantees of [Fair Food Data, 2026c]).

2. **Layer 1: Events and Telemetry.** The transition from raw observation to structured data. SenML [Jennings et al., 2018] encodes sensor measurements with units, time, and resource identifiers. GS1 EPCIS 2.0 [GS1, 2022b] encodes supply chain events (object, aggregation, transformation, association events).
3. **Layer 2: Domain Semantics.** The vocabulary that maps measurements and events into domain meaning. FoodOn [Dooley et al., 2018] provides the food matter ontology; SAREF4AGRI [ETSI, 2020b] provides the agricultural device ontology.
4. **Layer 3: Provenance Graph.** The relationships between entities, activities, and agents. PROV-O [W3C, 2013] provides the W3C standard graph model; BDT lifecycle operations map to PROV-O constructs.
5. **Layer 4: Credential Envelope.** The authenticated, signed wrapper around the data. W3C Verifiable Credentials 2.0 [W3C, 2024a] provides the credential structure with issuer, holder, verifier roles and cryptographic suites.
6. **Layer 5: Canonical Encoding.** The deterministic byte representation that produces the unique BDT identifier. Beeson [Tron et al., 2026] provides typed canonical serialisation and BMT hashing.
7. **Layer 6: Storage and Revocation.** The content addressed storage layer with retention economics and retrospective access revocation. Swarm [Trón, 2024] provides Single Owner Chunks, postage stamps, and DREAM revocation [Trón and Diedrich, 2025].
8. **Layer 7: Anchoring.** The public chain that records ownership transitions and dispute outcomes immutably. Ethereum [Wood, 2014] and Gnosis Chain [Gnosis, 2024] are the reference instantiations.

Direction of flow. Data flows upward: a sensor reading at Layer 0 is wrapped in SenML at Layer 1, semantically tagged at Layer 2, embedded in a provenance graph at Layer 3, signed as a Verifiable Credential at Layer 4, canonicalised at Layer 5, stored at Layer 6, and anchored at Layer 7. Verification flows downward: a verifier inspects an anchor at Layer 7, retrieves the canonical object at Layer 6, decodes its canonical form at Layer 5, verifies the credential at Layer 4, traverses the provenance graph at Layer 3, dereferences the domain terms at Layer 2, and inspects the underlying telemetry at Layer 1.

2.2 Selection Criteria

Each standard in the stack is selected against three criteria.

Open governance. The standard is published by a recognised standards body (W3C, GS1, IETF, ISO, OBO Foundry) with public review process and freely available specifications. The protocol does not depend on proprietary or restricted standards.

Production maturity. The standard has documented industrial deployments at scale relevant to the protocol’s target applications. Maturity reduces the risk of breaking changes and the cost of implementation.

Extensibility. The standard supports extension to new domains without breaking existing schemas. This is essential because the BDF protocol is multi vertical: today’s deployment focuses on freeze drying of food matter; tomorrow’s deployment may cover roasting, fermentation, brewing, extrusion, livestock, or aquaculture. The standards in the stack are chosen for their capacity to absorb new verticals without forcing a protocol revision.

2.3 Standards Selection Rationale

Table 1: Standards selection rationale. Maturity column: P = production deployment, R = recommendation status, E = emerging adoption. Extensibility column: H = high (designed for extension), M = medium (extensible with care), L = limited extension support.

Standard	Body	Maturity	Extensibility	Why selected
beeson	Tron et al.	E	H	Typed canonical encoding with BMT hashing; designed for content addressed storage
Swarm	EthSwarm	P	H	Postage TTL and DREAM revocation; cryptographic deletion mechanism
SenML (RFC 8428)	IETF	P	M	Compact sensor payload with units; widely deployed in IoT
EPCIS 2.0	GS1	P	M	Standard for supply chain event semantics; native to retail and food sectors
FoodOn	OBO	P	H	Hierarchical food ontology with cultivar and process descriptors
SAREF4AGRI	ETSI	R	H	Agricultural device and measurement ontology; aligned with EU initiatives
PROV-O	W3C	P	H	Standard provenance graph model; widely supported in scientific data infrastructure
VC 2.0	W3C	R	H	Latest credential standard with multi cryptosuite support and DID integration
Ethereum / Gnosis	community	P	M	Mature anchoring chains; settlement and dispute infrastructure

2.4 Standards Composition Rules

The standards are composed by the following rules:

1. **Layered without conflict.** Each standard operates on its layer. Lower layers are inputs to higher layers; higher layers do not modify lower layer data.

2. **Canonical at the BDT identity.** The unique identifier of any BDT is the BMT hash of its beeson canonical serialisation. All higher layer constructs (PROV-O nodes, VC envelopes, EPCIS events) reference the BDT by this identifier.
3. **Standards versions pinned per BDT.** The schema version of each standard used in a BDT is declared in the BDT's canonical metadata. Future versions of the standards do not retroactively invalidate existing BDTs.
4. **Selective disclosure preserves layering.** Encrypted payloads at Layer 6 do not affect Layer 7 (the anchor still records the BDT identity); decryption restores Layers 1 through 5 for the holder of the key.
5. **No rewriting.** The protocol does not modify standards. Where a standard is insufficient for a BDF use case (e.g. FoodOn lacks a specific cultivar IRI), the protocol either contributes upstream to the standard or extends through the standard's prescribed extension mechanism (e.g. a private OBO foundry namespace).

These rules guarantee that an integrator can adopt the BDF stack incrementally: implement Layer 1 first (SenML for sensors), then Layer 2 (FoodOn tagging), then Layers 3 and 4 (PROV-O and VC), then Layers 5 through 7 (beeson, Swarm, anchoring). Each layer adds capability without invalidating previous layers.

3 Canonical Encoding: beeson

The canonical encoding layer determines the unique identifier of every BDT and every signed payload in the protocol. This identifier is the BMT hash of a deterministic byte representation of the typed object. Beeson [Tron et al., 2026] provides the typed object notation and the canonicalisation rules.

3.1 Beeson Type System

Beeson is a typed object notation. Every value belongs to a declared type, and every type has a canonical serialisation derived from its definition. The type system supports:

- Primitive types: `string`, `int`, `float`, `bool`, `bytes`, `timestamp` (ISO 8601 UTC [International Organization for Standardization, 2019]), `geopoint` (latitude longitude in WGS 84), `hash` (32 byte BMT hash), `signature` (variable length depending on cryptosuite).
- Constrained primitives: `enum` from a documented set, `decimal` with declared precision, `string` with declared character set.
- Composite types: structures (typed attribute value matrices), arrays, maps, optional fields.
- Reference types: hash references to other beeson objects, providing a content addressed link.

Each typed object is a fully specified instance of a declared type. A schema is itself a typed object describing other types; schemas compose hierarchically and self describe.

3.2 Canonical Serialisation Rules

Beeson canonicalisation follows deterministic rules to ensure that two implementations produce identical byte sequences for the same logical object:

1. Field ordering by lexicographic name within a structure.
2. Numeric encoding without leading zeros, with a single canonical representation for integers and decimals.
3. String encoding in UTF-8 with rejection of non canonical or invalid sequences.
4. Boolean encoding as a single byte $\{0x00, 0x01\}$.
5. Timestamp encoding in ISO 8601 UTC with millisecond precision, no time zone offsets.
6. Geopoint encoding with declared decimal precision (default 6 digits, approximately 0.1 m at the equator).
7. Optional field encoding: present fields encode their value, absent fields are omitted from the byte sequence.
8. Array and map encoding with explicit length prefixes.
9. Hash and signature encoding as raw bytes (no hex or base64 wrapping at the canonical level).
10. Schema version field embedded as the first byte sequence of every typed object.

The conformance suite tests these rules across edge cases: negative zero, NaN values, leading and trailing whitespace, alternative UTF-8 encodings of the same character, and large integer overflow boundaries. Implementations that pass the conformance suite produce identical canonical bytes for the same logical input.

3.3 Binary Merkle Tree Hashing

The BMT hash $\mathcal{H}_{\text{BMT}}$ is computed over the canonical byte sequence by:

1. Splitting the byte sequence into chunks of fixed size (default 4 KiB).
2. Hashing each chunk with SHA-256 [NIST, 2015] to produce leaf nodes.
3. Constructing a binary Merkle tree over the leaves with SHA-256 as the internal compression function.
4. Returning the 32 byte root hash.

The BMT hash inherits the collision resistance of SHA-256. The tree structure additionally enables efficient retrieval and verification of arbitrary chunks: a verifier requesting a specific chunk receives the chunk plus a logarithmic proof of inclusion.

3.4 Schema Versioning

Each typed object carries a schema version in the first byte sequence of its canonical encoding. The version is an opaque hash reference to the canonical schema definition stored as another beeson object. Future schema revisions produce new version hashes; objects encoded under the old schema retain their original hash.

Backward compatibility. A schema revision that adds optional fields without changing existing field encodings is compatible: objects encoded under the old schema can be parsed by a parser that knows the new schema (the new optional fields are absent). The reverse is not always true; a parser that only knows the old schema may reject objects encoded under the new schema if non optional fields are added.

Migration. A deployment that upgrades to a new schema version creates new BDTs under the new version. Existing BDTs are not rewritten; the public chain ownership history of an old BDT remains valid because canonical hashes are stable. Higher tier consumers (e.g. verifiers) that require the new fields rely on the BDT's declared schema version to dispatch the appropriate parser.

3.5 Composition with Higher Layers

Beeson is the canonical encoding for the BDT itself and for every signed attestation vector. Higher layers (PROV-O, VC 2.0, EPCIS) reference BDT identifiers but operate on their own data models, which are then themselves represented as beeson typed objects when committed to the canonical record.

A pragmatic example: an EPCIS event is a JSON-LD document at Layer 1; for storage and identity, it is mapped to a beeson typed object whose schema includes the EPCIS event's required fields. The beeson object's canonical hash is the EPCIS event's BDT identifier in the protocol. The original JSON-LD document is preserved in the encoded bytes; tooling that prefers JSON-LD reads the beeson encoding and decodes back to JSON-LD, verifying the round trip.

3.6 Reference Implementation

The protocol uses the reference implementation of beeson released alongside [Tron et al., 2026]. The implementation provides:

- Canonical serialisation of typed objects.
- BMT hashing with chunk level proofs of inclusion.
- Schema definition language and validation.
- Round trip conversion to and from JSON, JSON-LD, and CBOR for compatibility with broader ecosystems.
- Conformance test suite covering the canonicalisation rules above.

A deployment shall pin the reference implementation version in its project design document. Updates that pass the conformance suite are non breaking and may be adopted at the deployment's discretion. Updates that change the canonicalisation rules are breaking and require a new schema version per Section 3.4.

4 Storage and Revocation: Swarm

The protocol stores the canonical BDT objects and signed attestation vectors on Swarm [Trón, 2024], a content addressed peer to peer storage network. Swarm provides the data availability, retention economics, and retrospective access revocation properties that the BDF protocol depends on.

4.1 Single Owner Chunks

A Single Owner Chunk (SOC) is a Swarm chunk whose address is derived from the owner’s public key plus a deterministic identifier, allowing the owner to publish updates at the same address over time. The protocol uses SOC’s to anchor mutable per BDT state, including the ownership chain head and the cumulative list of detector verdicts.

SOC address derivation. For a BDT with canonical hash H_O and a per object purpose tag (e.g. "transfer", "verdict"), the SOC address is

$$\text{SOC}_O = \mathcal{H}(H_O \parallel \text{purpose}),$$

where $\mathcal{H}$ is the SOC’s address hash function as defined in Swarm [Trón, 2024].

SOC update. The current owner publishes an update by signing the new content under their private key. The chunk is replicated across the SOC’s neighbourhood. Subsequent retrieval returns the latest valid update.

Conflicting updates. The owner cannot suppress an old update once it has been replicated; conflicting updates from the same key on the same SOC address are detectable by any verifier (consistent with the double signing argument of the security analysis [Fair Food Data, 2026c]).

4.2 Postage Stamps and Retention

Swarm chunks are persistent only while their postage stamps are valid. The postage stamp is an economic commitment by the uploader to compensate storage nodes for retention. When the stamp expires, the chunk is eligible for garbage collection.

Postage as retention TTL. The protocol uses postage as a retention mechanism: a deployment paying postage for its anchored data secures availability for the postage period. A deployment that ceases to renew postage signals retention non interest, equivalent to deletion at the storage layer.

Stamp economics. Postage cost grows linearly with the chunk count and the postage period. Reference parameters from [Trón, 2024] place the cost in the range of 1 to 10 BZZ tokens per terabyte month at typical network conditions. For typical BDF deployments the storage cost is bounded and predictable.

GDPR Article 17 compatibility. The postage TTL provides a structural mechanism for right to erasure: the operator ceasing to renew postage for a chunk causes the chunk to expire from the network. The cryptographic anchor on the public chain remains, but the encrypted plaintext becomes inaccessible. This is the policy independent erasure mechanism the protocol leverages.

4.3 Encryption Envelope

Swarm supports chunk level encryption under the BDF protocol’s selective disclosure mechanism. Each chunk is encrypted with a unique key, and the chunk reference embeds both the address and the decryption key.

Reference structure. A Swarm reference for an encrypted chunk is a tuple

$$\text{Ref} = (\text{addr}, \text{key}),$$

where *addr* is the chunk’s content address and *key* is the AES-256 encryption key. Without the key, the chunk content is indistinguishable from random bytes [Trón, 2024].

Selective disclosure. The chunk is replicated and stored encrypted. The owner shares the key selectively with parties who need access (a verifier under NDA, a regulator under enforcement powers, a buyer under contractual right). The disclosure event is itself a signed object on the public record, providing audit evidence of who received which key when.

Crypto shredding. The owner destroys the encryption key in their private system; the encrypted chunk remains in storage but is no longer decryptable. The destruction event is recorded as a signed object on the public chain, providing auditable evidence that erasure was exercised. This complements the postage TTL: postage TTL eliminates the chunk; crypto shredding renders it unintelligible while the chunk persists.

4.4 DREAM References

DREAM (Deletable References in Encrypted Append only Memory) [Trón and Diedrich, 2025] provides retrospective access revocation with quantitative cryptographic backing. The protocol uses DREAM references for high stakes selective disclosure cases where the right to revoke must remain effective even after the grantee has the reference.

Path based reference. A DREAM reference distributes the chunk identity across multiple network neighbourhoods. Following the path is required to assemble the decryption key. The path length n (typically 6) and the network honesty rate h together determine the revocation success probability.

Revocation by disruption. The uploader publishes new chunks to a designated batch that disrupts the path. The disruption succeeds if at least one neighbourhood on the path is honest and applies the

disruption. Quantitatively, the revocation probability is bounded by $1 - (1 - h)^n$, exceeding $1 - 10^{-6}$ at $h = 0.9$ and $n = 6$ (Section 8 of [Fair Food Data, 2026c]).

When to use DREAM versus crypto shredding. For low stakes selective disclosure (e.g. a buyer’s view of the recipe vector with a typical NDA), crypto shredding is sufficient. For high stakes cases (a contributor exercising the right to be forgotten with respect to data they contributed), DREAM provides quantitative cryptographic backing of the revocation. The protocol supports both and the deployment selects the appropriate mechanism per disclosure event.

4.5 Mapping BDF Primitives to Swarm

Table 2: Mapping of BDF protocol primitives to Swarm storage constructs.

BDF primitive	Swarm construct	Composition rule
BDT canonical object	Encrypted chunk(s)	Canonical hash H_O derives the chunk address; encryption key separately distributed
Ownership chain head	Single Owner Chunk	SOC address $SOC_O = \mathcal{H}(H_O \parallel \text{transfer})$
Detector verdict ledger	Single Owner Chunk	SOC address $\mathcal{H}(H_O \parallel \text{verdict})$
Distribution event	Encrypted chunk + SOC	Event payload chunk plus SOC update for parent BDT’s lifecycle
External Attestation	Encrypted chunk	Signed by laboratory; anchored to BDT via reference field in payload
Selective disclosure event	Signed object on chain	Logs key sharing; references encrypted chunk by Swarm reference
Right to erasure event	DREAM revocation chunk + signed log object	Triggers path disruption; signed log records the revocation as audit evidence

Reading the table. Every BDF primitive that requires storage maps to a specific Swarm construct. The composition rules guarantee that the canonical hash is preserved across encryption and that all updates are auditable through the SOC mechanism.

5 Sensor Telemetry: SenML

The Sensor Measurement Lists (SenML) standard [Jennings et al., 2018] encodes sensor measurements with units, timestamps, and resource identifiers in a compact, transport agnostic format. The protocol uses SenML for the lowest level telemetry payload entering the Agronomic, Logistic, and Process vectors.

5.1 SenML Record Structure

A SenML message is an array of measurement records. Each record carries a base name, a sensor identifier, a measurement value, a unit, and a timestamp. The standard supports value types for numeric, boolean, string, and binary measurements.

Listing 1: SenML record for a soil sensor reading. The base name and base time apply to all records in the array; per record fields override.

```
[
  {
    "bn": "urn:dev:bdt:soil:S1:",
    "bt": 1773774038,
    "bu": "Cel",
    "n": "temperature",
    "v": 18.7
  },
  {
    "n": "humidity",
    "u": "%RH",
    "v": 62.4
  },
  {
    "n": "conductivity",
    "u": "S/m",
    "v": 0.0021
  }
]
```

Field semantics.

bn	Base name. A URN identifying the device or sensor.
bt	Base time. Absolute time as Unix epoch seconds.
bu	Base unit. Default unit for records that omit u.
n	Record name. Appended to the base name to form the full identifier.
u	Per record unit (overrides bu).
v	Numeric value. Other type fields exist for boolean, string, binary.
t	Per record time offset relative to bt (optional).

5.2 Units and Measurement Types

SenML uses standard SI units with the IETF additional units extension [Bormann, 2020]. The protocol mandates the SI units for measurements covered by the four attestation vectors, summarised in Table 3.

Table 3: Required SenML units for measurements in BDF attestation vectors. The mapping is exact: every protocol level measurement carries its declared unit in the SenML payload.

Measurement	SenML unit	Vector
Temperature	Ce1	$\vec{A}, \vec{L}, \vec{P}$
Relative humidity	%RH	$\vec{A}, \vec{L}$
Soil pH	pH	$\vec{A}$
Electrical conductivity	S/m	$\vec{A}$
Mass	kg	$\vec{A}, \vec{L}, \vec{P}$
Pressure (chamber)	Pa	$\vec{P}$ (1 mTorr = 0.133 Pa)
Power consumption	W	$\vec{P}$
Cumulative energy	kWh	$\vec{P}$ (custom IETF additional unit)
Latitude / longitude	lat, lon	$\vec{A}, \vec{L}$
GPS speed	m/s	$\vec{L}$

Pressure unit conversion. Chamber pressure measurements in freeze drying are conventionally reported in mTorr, a non SI unit. SenML requires Pa in the canonical record; the protocol’s conversion is 1 mTorr = 0.133322 Pa. The original mTorr reading is preserved in the human readable view; the canonical SenML record stores Pa.

5.3 Time Anchoring

SenML uses Unix epoch seconds for absolute time. The protocol additionally requires that every signed payload carry an ISO 8601 UTC timestamp [International Organization for Standardization, 2019] alongside the SenML bt field. The dual representation allows IoT devices to publish in compact SenML form while higher tier verifiers consume the ISO 8601 string when needed for human readable display.

Bounded skew. Per A8 of the security analysis [Fair Food Data, 2026c], all signed payloads shall carry timestamps within a bounded skew $\Delta_t \leq 60$ seconds of the consensus time. A deployment that detects a SenML payload with timestamp outside the skew rejects the payload at the gateway.

5.4 Aggregation at the Gateway

Raw SenML streams from sensors are typically high frequency (60 second cadence for the BDF protocol per Section 6.3 of the methodology paper [Fair Food Data, 2026b]). The protocol does not commit every

raw record to the canonical record; it commits aggregates.

Aggregation rules.

- Per stage aggregates: mean, p05, p50, p95, max for each parameter within a freeze drying stage.
- Per segment aggregates for transport: distance, duration, max speed, p95 temperature within the segment.
- Per cohort aggregates for field sensors: mean and percentiles over a defined window (typically per agricultural block per day).

The aggregates are themselves SenML records, encoded with statistical operator names (e.g. n: "temperature.mean"). The full underlying time series is preserved in the operator's signed log and is producible upon verifier request, but is not in the canonical Process Vector.

Why aggregate. The aggregates are sufficient for T_2 envelope and conservation tests at the protocol level. The full time series serves T_3 classification (where sufficient resolution is required) and dispute resolution at T_4 . Storing only aggregates on chain and the time series in a signed log keeps the canonical record compact while preserving auditability.

6 Supply Chain Events: GS1 EPCIS 2.0

GS1 EPCIS (Electronic Product Code Information Services) version 2.0 [GS1, 2022b] is the standard for capturing and sharing supply chain events. It is the dominant standard for retail and food sector traceability and is required for compliance with FSMA Section 204 (US) Critical Tracking Events. The protocol uses EPCIS to encode supply chain events that correspond to BDT lifecycle operations.

6.1 EPCIS Event Types

EPCIS 2.0 defines five event types:

ObjectEvent

An action taken on one or more objects (observe, add, delete).

AggregationEvent

Objects associated with a parent (e.g. baskets aggregated into a pallet).

TransformationEvent

Input objects consumed and output objects produced (the protocol's burn semantics).

TransactionEvent

Objects associated with a business transaction (e.g. purchase, sale).

AssociationEvent

Objects associated with each other (e.g. a sensor associated with a container).

The protocol uses all five types. Table 4 maps BDT lifecycle operations to EPCIS event types.

Table 4: BDT lifecycle operations mapped to EPCIS event types.

BDT operation	EPCIS event	BizStep mapping (CBV)
Genesis (mint)	ObjectEvent	commissioning
Ownership transfer	TransactionEvent	shipping or receiving
Logistic segment	ObjectEvent	shipping, transporting, receiving
Cold chain attestation	AssociationEvent	sensor_reporting
Transformation (burn)	TransformationEvent	transforming
Distribution (fan-out)	AggregationEvent (inverse)	disaggregating
External Attestation issuance	ObjectEvent	inspecting
Detector verdict	ObjectEvent	sensor_reporting (custom subtype)

6.2 Core Business Vocabulary

The Core Business Vocabulary (CBV) [GS1, 2022a] provides controlled vocabularies for EPCIS event fields: BizStep (the business process step), Disposition (the state of the object after the event), and BizLocation (where the event occurred). The protocol uses CBV terms by default and extends with private namespace where the food sector specifically requires it.

BizStep extensions for freeze drying. The CBV does not currently include a step for freeze drying. The protocol uses the following extensions in the namespace `bdf:bizstep::`

- `bdf:bizstep:freeze_drying_loading`
- `bdf:bizstep:freeze_drying_primary_sublimation`
- `bdf:bizstep:freeze_drying_secondary_drying`
- `bdf:bizstep:freeze_drying_unloading`
- `bdf:bizstep:stabilisation_quality_check`

These are documented in the protocol’s reference vocabulary file and are submitted to GS1 for inclusion in a future CBV revision.

6.3 Identifiers: GLN, GTIN, EPC

EPCIS uses GS1 identifiers for parties, locations, and objects.

GLN. The Global Location Number [GS1, 2024] identifies parties and physical locations. Every BDF deployment registers GLNs for: the recovery point, the stabilisation facility, the storage facility, each transport carrier, and each laboratory issuing External Attestations.

GTIN. The Global Trade Item Number identifies the stabilised product as a trade item. Each distinct stabilised SKU has its own GTIN. The GTIN is associated with the BDT canonical hash through an `AssociationEvent`.

EPC. The Electronic Product Code identifies individual instances of an object. Each BDT instance has an EPC of the form `urn:bdf:bdt:canonical-hash`, providing a one to one mapping between EPCIS object identifier and BDF canonical hash.

6.4 EPCIS to BDT Bridge

An EPCIS event is a JSON-LD document. The protocol embeds the EPCIS event into a beeson typed object whose schema includes the EPCIS required fields plus the BDF specific extensions (signature, reference to the BDT canonical hash). The beeson object's BMT hash is the BDT canonical identifier of the event itself.

Listing 2: Excerpt of an EPCIS 2.0 `ObjectEvent` for a basket genesis attestation. The event is mapped to a beeson typed object whose canonical hash becomes the BDT identifier.

```
{
  "@context": "https://gs1.org/voc/",
  "type": "ObjectEvent",
  "eventID": "urn:uuid:5e6b...",
  "eventTime": "2026-03-17T19:00:38Z",
  "eventTimeZoneOffset": "+00:00",
  "epcList": [
    "urn:bdf:bdt:0xabc123..."
  ],
  "action": "ADD",
  "bizStep": "urn:epcglobal:cbv:bizstep:commissioning",
  "disposition": "urn:epcglobal:cbv:disp:active",
  "bizLocation": {
    "id": "urn:epc:id:sgln:0123456.0001.0"
  },
  "readPoint": {
    "id": "urn:epc:id:sgln:0123456.0001.RP1"
  },
  "extension": {
    "bdf:lotMassKg": 3.0,
    "bdf:cultivar": "Biloxi",
    "bdf:minterType": "humanoperator"
  }
}
```

Composition with the rest of the stack. The EPCIS event references the BDT canonical hash (0xabc123...) which is computed at the beeson layer. The event itself is signed at the credential layer (Section 10), embedded in a provenance graph at PROV-O (Section 9), and stored on Swarm (Section 4). The EPCIS payload remains intact through the layers, providing native compatibility with EPCIS conformant retail and food infrastructure.

6.5 FSMA Section 204 Compatibility

The US FDA’s Food Safety Modernization Act Section 204 mandates Critical Tracking Events (CTEs) and Key Data Elements (KDEs) for high risk foods. EPCIS 2.0 is the recommended standard for FSMA 204 compliance.

CTE coverage. The protocol’s BDT lifecycle operations cover the FSMA CTEs:

- Receiving event → `TransactionEvent` with `receiving` BizStep.
- Transformation event → `TransformationEvent` with `transforming` BizStep.
- Shipping event → `TransactionEvent` with `shipping` BizStep.
- Selective disclosure to FDA inspector → supported via the BDF protocol’s selective disclosure mechanism (Section 4.3).

KDE preservation. The BDF protocol preserves KDEs across the lifecycle through the T_1 canonical hash mechanism. A FSMA inspector with access to the public chain can trace any FDA regulated lot from production to retail.

7 Food Domain Ontology: FoodOn

FoodOn [Dooley et al., 2018] is the harmonised food ontology under the OBO Foundry. It provides a hierarchical class system for food matter, processing methods, organisms of origin, and quality descriptors. The protocol uses FoodOn to bind every BDT to canonical identifiers from the food domain rather than using free text strings.

7.1 FoodOn Class Hierarchy

FoodOn classes are identified by IRIs of the form `http://purl.obolibrary.org/obo/FOODON.NNNNNNN` where NNNNNNN is a numeric class identifier. The hierarchy includes:

- **Food matter classes.** Each cultivated species and cultivar has a class. Example: `FOODON_03413102` for blueberry.
- **Processing classes.** Each processing method has a class. Example: `FOODON_03460017` for freeze drying.
- **Organism classes.** For animal and plant origins, organism classes are inherited from `NCBITaxon`.
- **Quality classes.** Quality descriptors (e.g. “ripe”, “frozen”) are also classed for use in process descriptions.

Cultivar specificity. FoodOn supports cultivar level specificity for major food species. Where cultivar level identifiers are not yet present in FoodOn (e.g. a specific local cultivar), the protocol creates a private subclass under the relevant species class and contributes the new class upstream to FoodOn.

7.2 Mapping BDF Vectors to FoodOn

Table 5: Mapping BDF vector fields to FoodOn classes.

BDF field	FoodOn class	Composition rule
food_category (C1 to C6)	Aggregate FoodOn classes per category	Fixed mapping table per protocol revision
cultivar	Cultivar specific subclass under species class	Pinned IRI per BDT; mismatch is a T_2 flag
harvest_method	FoodOn process class	Optional but recommended
stabilisation_process	FoodOn process class (e.g. freeze drying)	Required for stabilised output BDT
packaging_class	FoodOn package class	Optional

Worked example. A BDT for a Biloxi blueberry lot stabilised by freeze drying carries the FoodOn IRIs as follows:

Listing 3: FoodOn IRIs in a BDT extension. The IRIs are pinned at genesis and immutable for the BDT lifetime.

```
{
  "bdf:foodonspecies": "http://purl.obolibrary.org/obo/FOODON03413102",
  "bdf:foodoncultivar": "http://purl.obolibrary.org/obo/FOODONBILOXIPRIVATE",
  "bdf:foodonprocess": "http://purl.obolibrary.org/obo/FOODON03460017",
  "bdf:foodonquality": "http://purl.obolibrary.org/obo/FOODONFRESH"
}
```

Why pinned IRIs. Free text fields (e.g. a cultivar name) are language and locale dependent and admit fraud through naming variations. FoodOn IRIs are unambiguous, language independent, and verifiable against a canonical ontology. A verifier consuming the BDT dereferences the IRI and arrives at the same class as any other verifier.

7.3 Tier 3 Classifier and FoodOn

The Tier 3 spectroscopic classifier outputs a FoodOn IRI as its predicted class. The verifier compares the predicted IRI to the BDT’s declared IRI; a mismatch is a Tier 3 fail signal. This binds the spectroscopic identity check to the canonical ontology rather than to a free text label, enabling cross deployment comparability and consistent verifier logic.

7.4 Process Descriptor Mapping

FoodOn includes a hierarchy of food processing methods. The BDF protocol’s stabilisation transformation maps to FoodOn classes:

— Freeze drying → FOODON.03460017

- Spray drying → FOODON_03460028
- Sun drying → FOODON_03460023
- Dehydration (general) → FOODON_03430101
- Roasting → FOODON_03430202
- Fermentation → FOODON_03430402

The mapping supports the multi vertical extensibility of the protocol: a deployment for a new stabilisation method (e.g. vacuum impregnation) declares its FoodOn IRI in the project design document and the BDT canonical schema picks it up.

7.5 Provenance of the FoodOn IRI

The FoodOn IRI declared in a BDT at genesis is itself a canonical reference subject to FoodOn’s release cycle. The protocol pins the FoodOn release version per BDT (the FoodOn version at the time of genesis); future FoodOn revisions do not retroactively change the BDT’s declared identity. Where FoodOn deprecates an IRI in a future release, the protocol’s reference implementation maintains a forward map from deprecated to current IRIs, allowing verifiers to dereference both.

8 Agricultural Device Ontology: SAREF4AGRI

SAREF (Smart Applications Reference) is the ETSI core ontology for connected devices [ETSI, 2020a]. SAREF4AGRI [ETSI, 2020b] is the agricultural extension that covers field sensors, weather stations, soil instruments, livestock devices, and agricultural process equipment. The protocol uses SAREF4AGRI to bind every IoT device in the field to a canonical device identity with a known capability profile.

8.1 SAREF Core Concepts

SAREF defines the core classes:

`saref:Device`

Any physical device in the deployment.

`saref:Sensor`

A device that produces measurements.

`saref:Actuator`

A device that performs actions on the environment.

`saref:Measurement`

An individual measurement record with a value, a unit, and a timestamp.

`saref:Property`

The physical property being measured (temperature, humidity, etc.).

`saref:UnitOfMeasure`

The unit in which the measurement is expressed.

SAREF4AGRI extends these with agricultural specific subclasses.

8.2 SAREF4AGRI Device Classes

The protocol uses the following SAREF4AGRI classes for devices in the BDF deployments documented in the main paper [Fair Food Data, 2026a].

Table 6: SAREF4AGRI device classes used in BDF deployments. Each device in the project’s IoT inventory carries the corresponding SAREF4AGRI class IRI.

Device class	IRI
Soil sensor	saref4agri:SoilSensor
Weather station	saref4agri:WeatherStation
Crop monitoring sensor	saref4agri:CropSensor
Tractor	saref4agri:Tractor
Irrigation system	saref4agri:IrrigationSystem
Livestock identifier	saref4agri:AnimalTag
Cold chain logger	saref:Sensor (subclass for cold chain in transport)
SCADA endpoint (plant)	saref:Device (with industrial profile extension)
Calibrated scale	saref:Sensor (with mass property)

8.3 Device Identity in BDF

Every device in the project’s IoT inventory has a SAREF4AGRI device record committed to the project’s BDF record at deployment time. The record includes:

- Device identifier (URN form, typically a UUID).
- SAREF4AGRI class IRI.
- Manufacturer and model.
- Calibration certificate reference (where applicable).
- Public key of the device (for signing measurements).
- Geofence (where applicable).
- Deployment date.
- Maintenance history (linked from a separate SOC for updates).

Why on chain device identity. Tying every measurement to a known, calibrated device with on chain identity prevents the trivial attack of injecting fabricated measurements from an unattributed source. A verifier reading a measurement signed by a device key looks up the device record and confirms it is registered in the project’s IoT inventory.

Listing 4: SAREF4AGRI device record for a soil sensor in the BDF pilot. The record is itself a beeson typed object signed at deployment.

```
{
  "@context": "https://saref.etsi.org/saref4agri/",
  "@id": "urn:bdf:device:soilsensor1",
  "@type": "saref4agri:SoilSensor",
  "saref:hasManufacturer": "ExampleSensorCo",
  "saref:hasModel": "SoilProbe-200",
  "saref:hasCalibrationCertificate": "0xdef456...",
}
```

```

"saref:hasGeolocation": {
  "@type": "GeoCoordinates",
  "latitude": 4.6533,
  "longitude": -74.1234
},
"saref:hasPublicKey": "0xa1b2c3d4...",
"saref:hasDeploymentDate": "2026-03-15T12:00:00Z",
"bdf:foodontarget": "FOODON03413102",
"saref:measures": [
  "saref:Temperature",
  "saref:Humidity",
  "saref:ElectricConductivity"
]
}

```

8.4 SAREF Measurements and BDF Vectors

Each SenML record (Section 5) corresponds to a `saref:Measurement` produced by a registered device. The mapping is direct:

- SenML `bn` field → device URN, dereferenceable to a SAREF4AGRI device record.
- SenML `n` field → `saref:Property` (e.g. `saref:Temperature`).
- SenML `u` field → `saref:UnitOfMeasure`.
- SenML `v` field → `saref:hasValue`.
- SenML `bt` or `t` field → `saref:hasTimestamp`.

The composition allows a verifier to traverse from a single signed measurement up to the device, the property, the unit, and the calibration record, reproducing the full provenance of the reading from sensor to BDT.

8.5 Multi Vertical Extensibility

SAREF and SAREF4AGRI are part of a family of extensions: SAREF4ENER (energy), SAREF4ENVI (environment), SAREF4WATR (water), SAREF4LIFT (lifting and material handling), and others. The BDF protocol’s multi vertical roadmap (Section 10 of [Fair Food Data, 2026a]) leverages these extensions:

- Cold chain logistics → SAREF4ENVI for ambient sensors.
- Plant operations → SAREF4ENER for energy meters at the SCADA layer.
- Brewery and fermentation → SAREF core with custom subclasses (forthcoming SAREF4FOOD extension).

The protocol is designed to absorb new SAREF extensions without changes to the canonical BDT object: device records are typed objects whose class IRI is the only field that distinguishes between extensions.

9 Provenance Graph: PROV-O

PROV-O [W3C, 2013] is the W3C Recommendation for representing provenance graphs. It defines three core class types (entity, activity, agent) and a vocabulary of relationships among them. The protocol uses PROV-O to express the full provenance graph of a BDT, providing a standard semantic layer over the BDF specific lifecycle operations.

9.1 PROV-O Core Classes

`prov:Entity`

A physical, digital, conceptual, or other kind of thing that has identity. In BDF, every BDT is a `prov:Entity`.

`prov:Activity`

Something that occurs over a period of time and acts upon or with entities. In BDF, every lifecycle operation (genesis, transfer, transformation, distribution) is a `prov:Activity`.

`prov:Agent`

Something that bears responsibility for an activity. In BDF, every signing party (operator, IoT device, laboratory, plant) is a `prov:Agent`.

The relationships among these classes encode the provenance graph: entities are derived from other entities through activities, and activities are attributed to agents.

9.2 Mapping BDT Lifecycle to PROV-O

9.3 Worked Example: Transformation Provenance

A transformation event consumes input BDTs O_A, O_B and produces an output BDT O_C . The PROV-O representation:

Listing 5: PROV-O JSON-LD encoding of a transformation event in the BDF protocol. The graph captures inputs, outputs, the activity, and the responsible agent.

```
{
  "@context": "http://www.w3.org/ns/prov-o",
  "@graph": [
    {
      "@id": "urn:bdf:bdtd:0xinputa",
      "@type": "prov:Entity",
      "prov:label": "Recovered Food Lot Basket A"
    },
    {
      "@id": "urn:bdf:bdtd:0xinputb",
      "@type": "prov:Entity",
      "prov:label": "Recovered Food Lot Basket B"
    },
    {
      "@id": "urn:bdf:bdtd:0xoutputc",
      "@type": "prov:Entity",
      "prov:label": "Stabilised Output Lot",
    }
  ]
}
```

Table 7: BDT lifecycle operations mapped to PROV-O constructs.

BDF construct	PROV-O class	Relationship pattern
BDT O	prov:Entity	The entity has identity H_O
Genesis (mint)	prov:Activity	prov:wasGeneratedBy from new BDT to genesis
Ownership transfer	prov:Activity	prov:wasAttributedTo BDT to new owner
Transformation (burn $\rightarrow$ output)	prov:Activity	prov:wasDerivedFrom output to inputs; prov:used activity to inputs
Distribution (parent $\rightarrow$ packages)	prov:Activity	prov:wasDerivedFrom package to parent for each output
Detector verdict	prov:Entity	prov:wasGeneratedBy verdict to detector activity; prov:wasAssociatedWith verdict to verifier agent
External Attestation	prov:Entity	prov:wasGeneratedBy attestation to laboratory analysis activity
Operator (party)	prov:Agent	prov:Person or prov:Organization or prov:SoftwareAgent subclass
IoT device	prov:Agent	prov:SoftwareAgent for autonomous Proplets

```

    "prov:wasDerivedFrom": [
      "urn:bdf:bdt:0xinputa",
      "urn:bdf:bdt:0xinputb"
    ],
    "prov:wasGeneratedBy": "urn:bdf:activity:transformationevent42"
  },
  {
    "@id": "urn:bdf:activity:transformationevent42",
    "@type": "prov:Activity",
    "prov:startedAtTime": "2026-04-12T08:00:00Z",
    "prov:endedAtTime": "2026-04-13T03:30:00Z",
    "prov:used": [
      "urn:bdf:bdt:0xinputa",
      "urn:bdf:bdt:0xinputb"
    ],
    "prov:wasAssociatedWith": "urn:bdf:agent:plantscada1"
  },
  {
    "@id": "urn:bdf:agent:plantscada1",
    "@type": "prov:SoftwareAgent",
    "prov:label": "Plant SCADA system"
  }
]
}

```

Reading the graph. The output entity is `wasDerivedFrom` both input entities and `wasGeneratedBy` the transformation activity. The activity is attributed to the SCADA agent, which is itself an entity with its own provenance (typically a software registration record).

9.4 Distribution Provenance

A distribution event creates several package outputs from a single parent. The provenance graph captures each package as derived from the parent, plus the conservation invariant on metadata quantities.

Listing 6: PROV-O encoding of a distribution event. Each package is derived from the parent through the same distribution activity.

```
{
  "@context": "http://www.w3.org/ns/prov-o",
  "@graph": [
    {
      "@id": "urn:bdf:bd:0xparent",
      "@type": "prov:Entity",
      "bdf:massKg": 50.0
    },
    {
      "@id": "urn:bdf:activity:distributionevent7",
      "@type": "prov:Activity",
      "prov:used": ["urn:bdf:bd:0xparent"]
    },
    {
      "@id": "urn:bdf:bd:0xpkg1",
      "@type": "prov:Entity",
      "bdf:massKg": 12.0,
      "prov:wasDerivedFrom": "urn:bdf:bd:0xparent",
      "prov:wasGeneratedBy": "urn:bdf:activity:distributionevent7"
    },
    {
      "@id": "urn:bdf:bd:0xpkg2",
      "@type": "prov:Entity",
      "bdf:massKg": 18.0,
      "prov:wasDerivedFrom": "urn:bdf:bd:0xparent",
      "prov:wasGeneratedBy": "urn:bdf:activity:distributionevent7"
    },
    {
      "@id": "urn:bdf:bd:0xremainder",
      "@type": "prov:Entity",
      "bdf:massKg": 20.0,
      "prov:wasDerivedFrom": "urn:bdf:bd:0xparent",
      "prov:wasGeneratedBy": "urn:bdf:activity:distributionevent7"
    }
  ]
}
```

The conservation invariant ($12 + 18 + 20 = 50$ kg) is checked at the protocol layer; PROV-O simply records the relationships.

9.5 Verifier Walks the Graph

A verifier inspecting a credit issuance retrieves the BDT for the credited lot and walks the PROV-O graph backwards through:

- Generation: which activity produced this BDT?
- Used: which entities did that activity consume as inputs?
- Attribution: which agent was responsible for the activity?
- Recursive walk: for each input entity, repeat.

The walk terminates at genesis events (entities with no input). The full graph is the provenance of the credited lot from harvest to issuance. The verifier can trace any field along the walk to its source attestation.

Reproducibility. A second verifier with access to the same public record arrives at an identical graph. The graph is canonical, not interpretive.

9.6 Composition with Other Layers

PROV-O nodes reference BDT identifiers (canonical hashes) and agent identifiers (DIDs or registered keys). The graph itself is a beeson typed object whose canonical hash is the graph identifier. The graph is signed at the credential layer (Section 10) by the responsible agent and stored on Swarm.

The composition allows the protocol to inherit PROV-O's tooling ecosystem (ProvBook, ProvStore, etc.) while preserving the BDF cryptographic guarantees on identity and signing.

10 Credential Envelope: W3C Verifiable Credentials 2.0

W3C Verifiable Credentials 2.0 [W3C, 2024a] is the W3C Recommendation for cryptographically verifiable claims. Each BDF attestation vector is wrapped in a Verifiable Credential, providing a standard envelope with issuer identity, holder relationship, claims payload, and cryptographic proof. The protocol uses VC 2.0 because it is the lingua franca of credentialed data exchange across regulators, registries, and decentralised infrastructure.

10.1 VC 2.0 Structure

A Verifiable Credential is a JSON-LD document with the following required structure:

`@context`

The JSON-LD context, including the W3C VC 2.0 context plus the BDF specific extensions.

`type`

The credential type, including `VerifiableCredential` plus a vector specific subtype (e.g. `BdfAgronomicVector`).

issuer

The DID [W3C, 2022] or other identifier of the issuer.

validFrom

ISO 8601 timestamp of credential validity start.

credentialSubject

The subject of the credential (the BDT or the attestation vector content).

proof

The cryptographic proof, conforming to the W3C Data Integrity specification [W3C, 2024b].

10.2 BDF Vector as Verifiable Credential

Listing 7: BDF Logistic Vector wrapped as a W3C Verifiable Credential 2.0. The credential is signed by the carrier's DID with EdDSA.

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://bdf.foodontology.org/contexts/logistic-vector-v1"
  ],
  "type": ["VerifiableCredential", "BdfLogisticVector"],
  "issuer": "did:web:carrier.example/keys/2",
  "validFrom": "2026-03-17T19:05:42Z",
  "credentialSubject": {
    "id": "urn:bdf:bdt:0xabc123...",
    "bdf:contentId": "260317140247-02-03-07",
    "bdf:lotId": "CAN-102",
    "bdf:segmentIndex": 1,
    "bdf:segmentOrigin": {
      "@type": "GeoCoordinates",
      "latitude": 4.6533,
      "longitude": -74.1234
    },
    "bdf:segmentDestination": {
      "@type": "GeoCoordinates",
      "latitude": 4.6101,
      "longitude": -74.0817
    },
    "bdf:segmentDistanceKm": 5.2,
    "bdf:segmentMode": "road",
    "bdf:segmentStart": "2026-03-17T19:00:38Z",
    "bdf:segmentEnd": "2026-03-17T19:05:42Z",
    "bdf:refrigerated": true,
    "bdf:massAtOriginKg": 2.0,
    "bdf:massAtDestinationKg": 2.79,
    "bdf:carrierGln": "0123456789013"
  },
  "proof": {
    "type": "DataIntegrityProof",
    "cryptosuite": "eddsa-rdfc-2022",
    "created": "2026-03-17T19:05:43Z",
    "verificationMethod": "did:web:carrier.example/keys/2",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3MvJjEX..."
  }
}
```

```
}  
}
```

Field interpretation. The credential subject contains the BDF specific fields. The credential itself is signed under the issuer’s DID using EdDSA [Josefsson and Liusvaara, 2017] or ECDSA secp256k1 (the protocol supports both via the cryptosuite field).

10.3 Decentralized Identifiers (DIDs)

The issuer field uses W3C Decentralized Identifiers [W3C, 2022]. The protocol supports the following DID methods:

`did:web`

DID resolved through HTTPS at a well known URL. Suitable for organisations with web presence.

`did:key`

DID encoded directly in the public key. Suitable for ephemeral or device level identities.

`did:ethr`

DID anchored on Ethereum. Suitable for organisations whose identity is tied to on chain governance.

`did:pkh`

DID derived from a blockchain account. Used by Proplet devices anchored on chain.

DID for Proplet attestation. A Proplet device’s DID is bound to its hardware root of trust (TPM 2.0 endorsement key), with the DID document published at deployment time and signed by the device manufacturer. The DID provides cryptographic identity for the device’s signed attestations, enabling the security analysis’s hardware substitution argument (Section 10 of [Fair Food Data, 2026c]).

10.4 Cryptographic Suites

The protocol supports the following Data Integrity cryptosuites [W3C, 2024b]:

- `eddsa-rdfc-2022`: EdDSA on Ed25519 with RDF Dataset Canonicalisation. Default for most signers.
- `ecdsa-rdfc-2019`: ECDSA on secp256k1 with RDF Dataset Canonicalisation. Used where Ethereum/Gnosis chain integration is direct.
- `bbs-2023`: BBS+ signatures for selective disclosure use cases requiring zero knowledge.

Choice of cryptosuite. The default for new deployments is `eddsa-rdfc-2022` for performance and resistance to side channel attacks. ECDSA secp256k1 is selected when the issuer’s identity is anchored on Ethereum or Gnosis Chain, allowing the same key to sign on chain transactions and off chain credentials. BBS+ is selected for selective disclosure use cases where the holder must prove a property of the credential without revealing the entire payload.

10.5 Verifier Logic

A verifier consuming a Verifiable Credential performs the following sequence:

1. Parse the credential JSON-LD and validate against the W3C VC 2.0 context.
2. Resolve the issuer's DID document to retrieve the verification method.
3. Validate the proof using the cryptosuite specified in the proof block.
4. Apply BDF specific validation: schema version check, BDT identifier consistency, vector specific fields.
5. Pass the credential subject up to the BDF higher tier verifiers (T2, T3, T4 detectors).

Reproducibility. Verifier logic is deterministic and reference implemented. Any verifier with the same VC and DID document arrives at the same accept or reject verdict.

10.6 Selective Disclosure with Verifiable Credentials

VC 2.0 supports selective disclosure through derived credentials and BBS+ proofs. The protocol uses these mechanisms to allow holders to disclose specific fields of a credential without revealing the entire content.

Use case: Recipe vector disclosure to verifier. The plant operator issues a Recipe Vector credential with the full recipe content. The verifier under NDA receives a derived credential that proves the recipe's commitment hash matches the on chain anchor without revealing the recipe plaintext beyond the fields the verifier is authorised to inspect.

Use case: Carbon credit buyer disclosure. A buyer of credits issued under the BDF methodology may receive a derived credential showing the issuance event's signature and the project's T_2 verdict aggregates without exposing per lot recipe data. The buyer's view is bounded by the contractual scope of the disclosure event (Section 4.3).

10.7 Status and Revocation

VC 2.0 supports revocation through the `credentialStatus` field. The protocol uses revocation for:

- Withdrawn detector verdicts: a verdict that is later determined to be in error is revoked through the credential status mechanism, with the revocation event itself signed and anchored.
- Compromised issuer keys: an issuer whose key has been compromised revokes all credentials issued after the compromise window through a status list.

The revocation event is itself a Verifiable Credential, providing audit trail integrity. The credential status check is part of the verifier's standard validation sequence (Section 10.5).

11 Public Chain Anchoring

The protocol anchors specific events to a public chain to obtain immutable timestamping, conflict resolution, and Sybil resistance for ownership transitions. This section documents which events are anchored, why, and on which chains.

11.1 What Gets Anchored

The protocol anchors a minimal set of events to the public chain. Bulk data lives on Swarm; the chain serves as a coordination and conflict resolution layer.

Table 8: Events anchored to the public chain. Each anchor is a minimal commitment to the canonical hash of the underlying signed object on Swarm.

Event	Chain commitment	Purpose
BDT genesis	H.O reference	Establish first existence and ownership
Ownership transfer	SOC update reference	Public conflict detection on transfers
Burn (terminal transfer to TransformAddr)	SOC final update	Mark BDT consumed
Distribution event	Event hash + parent BDT reference	Establish fan-out lineage
External Attestation	H.E reference	Anchor laboratory evidence with timestamp
Detector verdict	Verdict hash	Public record of T2/T3/T4 verdicts
Calibration correction event	Event hash	Audit trail of bias corrections
Selective disclosure event	Event hash	Audit trail of key sharing
Revocation event (DREAM)	Revocation hash	Audit trail of erasure exercise
Dispute filing and outcome	Hash references	Public record of dispute lifecycle

What does not get anchored. The bulk of attestation vector content lives on Swarm, encrypted, with only its canonical hash on chain. This includes: per measurement SenML records, per cycle Process Vector telemetry, per segment Logistic Vector data, full Recipe Vector plaintext, External Attestation bodies. The chain commits to the integrity of these objects without storing them.

11.2 Reference Chains

The protocol's reference deployment supports two chains.

Ethereum. Ethereum [Wood, 2014] is the original BDF reference chain. It provides the strongest decentralisation and adoption profile, with extensive tooling and registries.

Gnosis Chain. Gnosis Chain [Gnosis, 2024] is the secondary reference chain, with EVM compatibility and lower transaction costs. The protocol supports anchoring on Gnosis Chain for deployments where transaction frequency is high (e.g. per lot anchoring across thousands of lots per day).

Cross chain consistency. A deployment may anchor on multiple chains simultaneously for redundancy. Cross chain anchors are linked through their commitment of the same canonical hash; consistency is structural, not consensus dependent.

11.3 Smart Contract Surface

The protocol's on chain footprint is intentionally minimal. The reference contracts are:

- **BDT Registry.** Maps BDT canonical hashes to their genesis owner and current SOC update reference. Any party can read; the genesis owner writes the registry entry.
- **Ownership Transfer.** Records ownership transfers as signed objects committing to BDT hash and new owner. Conflicts (double signing) are publicly detectable on inspection.
- **External Attestation Anchor.** Records the canonical hash of an External Attestation issued by a registered laboratory.
- **Verdict Anchor.** Records detector verdicts at T_2, T_3, T_4 .
- **Dispute Resolution.** Records dispute filings, mediator reports, and outcomes as signed objects.

The contracts do not perform business logic validation; they are write append registries with read functions for verifiers. Business logic (the Tier 2 detectors, the ER calculation, the verifier validation) runs off chain over the signed inputs.

Why minimal. The decision to keep on chain logic minimal serves three goals: (i) lower gas cost for deployments at scale, (ii) avoid coupling protocol semantics to a specific chain's evolution, (iii) preserve clean separation between the data integrity layer (cryptographic) and the business validation layer (statistical and forensic).

11.4 Settlement vs Anchoring

The protocol distinguishes anchoring (the cryptographic commitment to a hash) from settlement (the transfer of value associated with the anchored event).

Anchoring is mandatory. Every event in Table 8 is anchored to a chain. The anchor is the public proof that the event occurred and references a specific canonical object.

Settlement is optional. Some events have associated value transfers (e.g. a credit retirement payment, a stake slashing event). Where settlement occurs on the same chain as the anchor, the protocol's reference contracts provide hooks for the value transfer. Where settlement occurs on a different chain or off chain, the protocol's anchor is the immutable record of the event regardless of where settlement happens.

11.5 Gas Cost and Throughput

The protocol's anchoring transactions are small: typically a single 32 byte hash plus minimal metadata. Per transaction gas cost is dominated by the chain's base fee plus a constant for storage write. Reference figures for the two supported chains:

Ethereum.

At typical gas prices (10 to 50 gwei), a BDT genesis anchor costs approximately 0.3 to 1.5 USD equivalent. A deployment with 1,000 BDTs per day faces 300 to 1500 USD per day in anchoring cost on Ethereum.

Gnosis Chain.

Gas cost is roughly 1 percent of Ethereum's, making per BDT anchor cost a fraction of a cent. Suitable for high frequency anchoring.

Batching. The protocol's reference contracts support batched anchoring: multiple BDTs can be anchored in a single transaction by supplying a list of canonical hashes. Batching reduces per BDT cost by a factor of 5 to 10 in typical use.

11.6 Chain Migration and Multi Chain Deployments

A deployment may need to migrate from one chain to another (e.g. a Gnosis Chain deployment moving to Ethereum for regulatory reasons, or vice versa for cost reasons). The protocol supports migration through:

- **Re-anchoring.** The deployment commits the canonical hashes of historical BDTs to the new chain, preserving provenance under both chains during a transition window.
- **Multi chain anchors.** New BDTs may be anchored to multiple chains simultaneously, providing redundancy against any single chain's consensus failure.
- **Bridge contracts.** Where the two chains have a trusted bridge, the protocol uses the bridge to mirror anchor events; otherwise the deployment publishes independently to each chain.

Verifier behaviour during migration. A verifier inspecting BDTs across the migration window resolves anchors on both chains and treats the BDT as valid if either chain's anchor verifies. The reference implementation handles multi chain anchor resolution transparently.

12 Vector Schema Mappings

This section consolidates the per vector mappings from the abstract attestation vectors of the BDF main paper [Fair Food Data, 2026a] to the concrete standards. Each subsection takes one vector and tabulates its required fields against the standards layers, providing the complete reference an integrator needs to implement the vector.

12.1 Agronomic Vector $\vec{A}$

Table 9: Agronomic Vector field mapping across the standards stack.

Field	Source standard	Type	Notes
lot_id	Internal	string	Unique within project
content_id	Internal	string	Cross reference identifier
recovery_point	ISO 19115 / GeoJSON	geopoint	WGS 84 lat/lon
recovery_date	ISO 8601	timestamp	UTC
mass.at_recovery_kg	ISAREF4AGRI	float	Property = mass, unit = kg
food_category	FoodOn	enum + IRI	C1 to C6 plus pinned IRI
cultivar	FoodOn	IRI	Cultivar specific subclass
minter_type	Internal	enum	Trust profile
operator_id	W3C DID	DID	Signer identity
soil_sensor_data	SenML	array	Per device per measurement
weather_data	SenML	array	Per measurement
harvest_method	FoodOn	IRI	Optional process descriptor
block / parcel_id	GS1 GLN	string	Encodes farm location
epcis_event	GS1 EPCIS 2.0	object	ObjectEvent with bizStep = commissioning
prov_node	PROV-O	object	prov:Entity with prov:wasGeneratedBy
vc_envelope	W3C VC 2.0	object	Type = BdfAgronomicVector
signature	EdDSA / ECDSA	signature	Cryptosuite per device

12.2 Logistic Vector $\vec{L}$

12.3 Process Vector $\vec{P}$

12.4 Recipe Vector $\vec{R}$

Recipe encryption. The Recipe Vector is the canonical confidential payload. The plaintext is encrypted at the Swarm chunk layer; the canonical hash is committed to the public chain unencrypted; the encryption key is shared selectively under NDA with verifiers and under contract with buyers (Section 4.3).

12.5 External Attestation $\vec{E}$

Laboratory key registration. The laboratory's signing key is registered alongside its ISO 17025 accreditation number in the project's BDF record at deployment. A verifier consuming an External Attestation looks up the laboratory's registration to confirm A7 (Section 3 of [Fair Food Data, 2026c]).

Table 10: Logistic Vector field mapping across the standards stack.

Field	Source standard	Type	Notes
segment_index	Internal	int	Per lot segment counter
segment_origin	ISO 19115	geopoint	WGS 84 lat/lon
segment_destination	ISO 19115	geopoint	WGS 84 lat/lon
segment_distance_km	Computed	float	From origin and destination
segment_mode	Internal	enum	road, rail, air, sea
segment_start	ISO 8601	timestamp	UTC
segment_end	ISO 8601	timestamp	UTC
refrigerated	Internal	bool	Cold chain flag
setpoint_temp_c	SAREF / SenML	float	Optional, for cold chain
observed_temp_p95	SAREF / SenML	float	p95 over segment
mass_at_origin_kg	SAREF4AGRI	float	Calibrated scale
mass_at_destination_kg	SAREF4AGRI	float	Calibrated scale
carrier_id	GS1 GLN	string	Carrier party
driver_id	W3C DID (optional)	DID	Operator identity
epcis_event	GS1 EPCIS 2.0	object	ObjectEvent or Transaction-Event
prov_node	PROV-O	object	prov:Activity for transit
vc.envelope	W3C VC 2.0	object	Type = BdfLogisticVector
signature	EdDSA / ECDSA	signature	Carrier or device key

Table 11: Process Vector field mapping across the standards stack.

Field	Source standard	Type	Notes
cycle_id	Internal	string	Plant scoped
facility_id	GS1 GLN	string	Plant location
cycle_start	ISO 8601	timestamp	UTC
cycle_end	ISO 8601	timestamp	UTC
input_lots	Internal	array	Each with lot_id and mass_kg
output_lots	Internal	array	Each with lot_id and stabilised_mass_kg
total_water_removed_kg	Computed	float	Conservation invariant
total_electricity_senML	SenML / SAREF	float	Cumulative cycle energy
recipe_vector_id	Internal	hash	Reference to $\vec{R}$
stages	SenML aggregates	array	Per stage telemetry aggregates
refrigerant_state	Internal	object	Designation, charge, GWP
underlying_log_anchor	Internal	hash	Reference to full time series log
epcis_event	GS1 EPCIS 2.0	object	TransformationEvent
prov_node	PROV-O	object	prov:Activity for the cycle
vc.envelope	W3C VC 2.0	object	Type = BdfProcessVector
signature	EdDSA / ECDSA	signature	SCADA system key

13 Worked Example: One Reading Through the Stack

This section walks a single physical observation through every layer of the standards stack, from the field IoT device to the Swarm anchor and the public chain commitment. The example is grounded in

Table 12: Recipe Vector field mapping across the standards stack.

Field	Source standard	Type	Notes
recipe_id	Internal	string	Identifier
cycle_id	Internal	string	Bound to a future cycle
declared_inputs	Internal	array	lot_id and expected_mass_kg
expected_yield_kg	Internal	float	Stabilised output target
expected_energy_kwh	Internal	float	Energy budget
refrigerant	Internal	object	Type and charge declared
stage_targets	Internal	array	Per stage target envelopes
global_tolerances	Internal	object	Allowable deviations
operator_id	W3C DID	DID	Plant operator identity
epcis_event	GS1 EPCIS 2.0	object	ObjectEvent commitment
prov_node	PROV-O	object	prov:Plan entity
vc_envelope	W3C VC 2.0	object	Type = BdfRecipeVector
encryption	Swarm chunk encryption	object	Per chunk key for selective disclosure
signature	EdDSA / ECDSA	signature	Operator key

Table 13: External Attestation field mapping across the standards stack.

Field	Source standard	Type	Notes
attestation_id	Internal	hash	Canonical hash
bdt_reference	Internal	hash	BDT being attested
lot_id	Internal	string	Lot scoped
laboratory_id	GS1 GLN	string	Laboratory location and party
iso17025_accreditation_number	ISO 17025	string	Accreditation number
iso17025_accreditation_body	ISO 17025	string	Accrediting body name
method_standard	ISO method registry	string	Applied method
sample_collection_iso8601	ISO 8601	timestamp	UTC
analysis_date	ISO 8601	timestamp	UTC
measurement_class	Internal	enum	composition, isotopic, etc.
measurement	Internal + units	object	Class specific structured payload
uncertainty_pct	ISO 17025	float	Reported by laboratory
epcis_event	GS1 EPCIS 2.0	object	ObjectEvent with bizStep = inspecting
prov_node	PROV-O	object	prov:Entity for the attestation
vc_envelope	W3C VC 2.0	object	Type = BdfExternalAttestation
signature	EdDSA / ECDSA	signature	Laboratory key

real pilot data from the BDF main paper [Fair Food Data, 2026a].

The observation. On 17 March 2026 at 19:00:38 UTC, a soil sensor identified as `soil_sensor_1` located at the cultivation block of the pilot project measured a soil temperature of 18.7 degrees Celsius, soil humidity of 62.4 percent, and soil electrical conductivity of 0.0021 S/m. This is one of approximately 130 readings produced by this sensor over the seven week observation window (Section 7.2 of [Fair Food Data, 2026a]).

13.1 Layer 0: Physical Reality

The sensor probe is buried at 30 cm depth in cultivation block B-01. Three calibrated transducers measure temperature, humidity, and electrical conductivity. The sensor's microcontroller reads each transducer at the configured cadence (60 second intervals) and prepares the readings for transmission.

13.2 Layer 1: SenML Encoding

The sensor's firmware encodes the three measurements into a SenML record:

```
[
  {
    "bn": "urn:bdf:device:soilsensor1:",
    "bt": 1773774038,
    "n": "temperature",
    "u": "Cel",
    "v": 18.7
  },
  {
    "n": "humidity",
    "u": "%RH",
    "v": 62.4
  },
  {
    "n": "conductivity",
    "u": "S/m",
    "v": 0.0021
  }
]
```

The SenML payload is transmitted from the sensor to the field gateway via LoRaWAN. The gateway aggregates per minute readings from all field sensors into a daily SenML batch.

13.3 Layer 2: Domain Semantics (FoodOn + SAREF4AGRI)

The gateway annotates the SenML batch with domain semantics:

- Each device URN dereferences to a SAREF4AGRI device record (Section 8.2). `soil_sensor_1` is registered as `saref4agri:SoilSensor` with manufacturer, model, calibration certificate, and public key.
- Each measurement property dereferences to a SAREF property: `saref:Temperature`, `saref:Humidity`, `saref:ElectricConductivity`.
- The cultivation block carries a FoodOn cultivar IRI, in this pilot `FOODON_BILOXI_PRIVATE` mapped to `FOODON_03413102` (blueberry).

The annotated batch is now self describing in the food and agricultural domain.

13.4 Layer 3: PROV-O Provenance Graph

The gateway constructs a PROV-O fragment for the daily aggregate:

```
{
  "@context": "http://www.w3.org/ns/prov-o",
  "@graph": [
    {
      "@id": "urn:bdf:measurementbatch:soilsensor1:2026-03-17",
      "@type": "prov:Entity",
      "prov:wasGeneratedBy": "urn:bdf:activity:sensorreporting2026-03-17",
      "prov:wasAttributedTo": "urn:bdf:device:soilsensor1"
    },
    {
      "@id": "urn:bdf:activity:sensorreporting2026-03-17",
      "@type": "prov:Activity",
      "prov:startedAtTime": "2026-03-17T00:00:00Z",
      "prov:endedAtTime": "2026-03-18T00:00:00Z",
      "prov:wasAssociatedWith": "urn:bdf:device:soilsensor1"
    }
  ]
}
```

13.5 Layer 4: W3C Verifiable Credential Envelope

The gateway wraps the SenML batch + domain annotations + PROV-O graph into a Verifiable Credential. The gateway is the issuer (its DID is `did:web:gateway.example/keys/1`); the credential subject is the daily measurement batch.

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://bdf.foodontology.org/contexts/agronomic-vector-v1"
  ],
  "type": ["VerifiableCredential", "BdfAgronomicMeasurement"],
  "issuer": "did:web:gateway.example/keys/1",
  "validFrom": "2026-03-18T00:00:01Z",
  "credentialSubject": {
    "id": "urn:bdf:measurementbatch:soilsensor1:2026-03-17",
    "bdf:device": "urn:bdf:device:soilsensor1",
    "bdf:senml": "...",
    "bdf:provGraph": "...",
    "bdf:foodontarget": "FOODON03413102"
  },
  "proof": {
    "type": "DataIntegrityProof",
    "cryptosuite": "eddsa-rdfc-2022",
    "created": "2026-03-18T00:00:02Z",
    "verificationMethod": "did:web:gateway.example/keys/1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3Mvq..."
  }
}
```

The credential is cryptographically signed under the gateway's Ed25519 key.

13.6 Layer 5: Beeson Canonical Encoding

The Verifiable Credential is now canonicalised under beeson [Tron et al., 2026] into its typed form. Beeson:

1. Parses the JSON-LD into a typed object tree.
2. Orders fields lexicographically.
3. Encodes timestamps in ISO 8601 UTC, geopoints with 6 digit precision, etc.
4. Embeds the schema version as the first byte sequence.
5. Produces a unique canonical byte sequence.

The byte sequence is then BMT hashed:

$$HO = 0x9f8a52b30c4d1e7f6a8c2d3e9b4f7a1d0e5c6b8a3f9d2e4c1b7a8d6f5e3c2b4a$$

This 32 byte hash is the canonical identifier of the BDT for this daily measurement batch. Any party computing the canonical encoding from the same source data arrives at the same hash.

13.7 Layer 6: Swarm Storage

The canonical encoded byte sequence is uploaded to Swarm [Trón, 2024]. The bytes are split into 4 KiB chunks; each chunk is encrypted with a unique AES-256 key. The chunk references (address plus key) are returned to the gateway.

The gateway computes the SOC address for this BDT's ownership chain:

$$SOCO = H(HO \text{ — "transfer"}) = 0x4b7e9a0c\dots$$

It publishes the initial ownership record to this SOC, signed under its DID key. Postage is paid to retain the chunks for a configured period (typically aligned with the project's crediting period plus retention buffer).

13.8 Layer 7: Public Chain Anchoring

The gateway submits an anchor transaction to Gnosis Chain (the project's selected anchor chain). The transaction calls the BDT Registry contract with:

- Canonical hash $H_O = 0x9f8a52b3\dots$
- SOC update reference.
- Genesis owner address (the gateway's Ethereum address).
- Timestamp.

The transaction is mined and finalised. Gas cost is approximately 0.005 USD equivalent at typical Gnosis Chain gas prices. The on chain entry is now public, immutable, and timestamped.

13.9 End to End Verification

A verifier inspecting this measurement batch reverses the walk:

1. Reads the BDT Registry on Gnosis Chain to retrieve the canonical hash H_O and the SOC reference.
2. Retrieves the SOC update from Swarm to obtain the current ownership chain head.
3. Retrieves the encrypted chunks from Swarm, requesting decryption keys under the project's selective disclosure scope.
4. Reconstructs the canonical byte sequence and verifies that its BMT hash equals H_O .
5. Decodes the byte sequence to the Verifiable Credential.
6. Validates the credential signature against the issuer's DID document.
7. Walks the PROV-O graph to confirm the measurement's provenance.
8. Reads the SenML records, dereferences SAREF and FoodOn IRIs, and confirms the measurement is consistent with the project's expected envelope.

The full chain of evidence from physical sensor to public chain anchor is reproducible by any verifier with access to the public record, demonstrating the protocol's structural property: *any verifier with access to the public record can independently retrace the chain of evidence and arrive at the same verdict* (Section 1 of [Fair Food Data, 2026a]).

13.10 Summary of Layers Crossed

Table 14: The single sensor reading's representation at each layer of the stack.

Layer	Standard	Representation
0	Physical reality	Three transducer readings at 30 cm soil depth
1	SenML	JSON array with bn, bt, n, u, v fields
2	FoodOn + SAREF4AGRI	Annotated SenML with device IRIs and food cultivar IRI
3	PROV-O	JSON-LD graph with prov:Entity and prov:Activity
4	W3C VC 2.0	Signed credential with EdDSA proof and gateway DID issuer
5	beeson + BMT	Canonical byte sequence and 32 byte hash
6	Swarm	Encrypted chunks plus SOC ownership record
7	Gnosis Chain	Public anchor transaction recording H_O

The same observation, propagated through every layer, is now an auditable, verifiable, deletable record on the BDF protocol.

14 Versioning and Migration

The BDF stack composes nine open standards, each with its own release cycle. A deployment that runs for years across multiple crediting periods will encounter standards revisions, deprecations, and occasional breaking changes. This section documents the versioning policy and the migration strategy.

14.1 Standards Version Pinning

Every BDT carries the version of each standard used in its construction. The pinning is recorded in the BDT’s canonical metadata at genesis and is immutable for the BDT’s lifetime.

Table 15: Standards pinned per BDT at genesis. The fields are part of the canonical schema and cannot be changed retroactively.

Standard	Version field	Default
beeson	bdf:beesonVersion	1.0
Swarm	bdf:swarmVersion	Bee 2.x
SenML	bdf:senmlVersion	RFC 8428
GS1 EPCIS	bdf:epcisVersion	2.0
GS1 CBV	bdf:cbvVersion	2.0
FoodOn	bdf:foodonRelease	2025 quarterly release
SAREF4AGRI	bdf:saref4agriVersion	ETSI TS 103 410-6 v1.x
PROV-O	bdf:provOVersion	W3C 2013 Recommendation
W3C VC	bdf:vcVersion	2.0
W3C DID	bdf:didVersion	1.0
Anchoring chain	bdf:anchorChain	gnosis or ethereum mainnet

14.2 Compatible vs Breaking Revisions

Compatible revisions. A revision that adds optional fields, deprecates fields without removing them, or extends an enumeration is compatible. Existing BDTs continue to verify under their pinned version, and new BDTs may use the new fields. The protocol’s reference implementation supports parsers for the current version plus a configurable set of past versions.

Breaking revisions. A revision that removes required fields, changes field semantics, or modifies the canonical encoding is breaking. New BDTs use the new version; existing BDTs remain valid under the old version. The protocol does not retroactively rewrite BDTs across breaking revisions.

Verifier behaviour. A verifier consulting a BDT determines the BDT’s pinned standard version and dispatches the appropriate parser. The verifier does not assume the most recent version of any standard. This guarantees that BDTs created under historical standard versions remain verifiable indefinitely (subject to A1 cryptographic primitive validity, which has its own migration policy).

14.3 Cryptographic Primitive Migration

The cryptographic primitives (hash, signature, encryption) sit below the standards layers and have their own migration concerns. The protocol’s policy:

- **Hash function.** BDTs commit to BMT over SHA–256 at genesis. A future migration to a different hash function (e.g. SHA–3 or a post quantum candidate) requires a beeson schema

version field that distinguishes the hash family. Reference implementations support multiple hash families during transition.

- **Signature scheme.** BDTs sign with the cryptosuite declared in the credential proof block. Adding a new cryptosuite (e.g. Falcon for post quantum) does not invalidate existing signatures; verifiers know the cryptosuite from the proof block and dispatch accordingly.
- **Encryption.** Per chunk encryption keys are independently managed; rotating to a new symmetric primitive (e.g. AES-GCM-SIV) is a per chunk decision and does not affect previously encrypted chunks.

Post quantum readiness. The protocol’s reference implementation provides a hybrid signature mode: every signed payload is countersigned with both ECDSA secp256k1 and a post quantum candidate (currently Dilithium). This raises signature footprint but provides forward security against future cryptanalytic breaks. Deployments at high stakes adopt hybrid signatures preemptively.

14.4 FoodOn Class Evolution

FoodOn releases quarterly with new classes added and occasional deprecations. The protocol pins the FoodOn release version at BDT genesis. Specific concerns:

- **New classes.** A new FoodOn class added in a later release does not invalidate BDTs that used a parent class at the time of genesis. The BDT’s pinned IRI remains valid.
- **Deprecated classes.** FoodOn deprecates classes through obsolescence rather than deletion. The deprecated IRI continues to dereference; consumer software shall recognise the deprecation flag and prefer the replacement IRI for new BDTs.
- **Cultivar contributions.** Where a project introduces a new cultivar via a private subclass and later contributes it upstream, the upstream IRI replaces the private IRI for new BDTs while the historical BDTs retain the private IRI with a documented mapping.

14.5 EPCIS and CBV Vocabulary Updates

GS1 publishes annual updates to EPCIS and CBV. Most updates are compatible (new BizSteps, new Disposition values). Breaking updates are rare and typically come with a transition window of at least 12 months.

Custom BizStep extensions. The protocol’s freeze drying BizStep extensions (Section 6.2) live in a private namespace until adopted upstream by GS1. The transition from private to GS1 namespace updates the URN for new BDTs; historical BDTs retain the private URN.

14.6 Migration Procedure

When a deployment migrates to a new version of any standard:

1. Document the migration in the project design document with the source and target versions.

2. Update the project’s BDF stack configuration to use the new version for new BDTs.
3. Verify backward compatibility against the cohort of historical BDTs using the reference implementation.
4. Notify the registry of the migration as part of the next vintage year verification.
5. Continue to support parsers for the previous version for the duration of historical BDT verifiability.

Registry recognition. ICR, Verra, and Gold Standard recognise standards version migrations as routine project amendments rather than methodology revisions, provided the migration does not affect the protocol’s structural guarantees. The methodology paper [Fair Food Data, 2026b] documents the registry recognition process.

14.7 Deprecation of the BDF Stack Itself

The BDF protocol may itself revise its standards selection over time. A future release may replace one of the current standards with a more mature alternative or absorb a new standard for a previously unaddressed concern.

Forward compatibility. BDTs created under the current stack remain valid indefinitely under their pinned versions. A future BDF release may use a different stack composition; the protocol’s reference verifier dispatches based on the BDT’s declared stack version.

Sunset windows. The protocol does not unilaterally sunset historical BDTs. Where a primitive becomes unsafe (e.g. a hash function break), affected BDTs may need re-anchoring under the new primitive; the migration is at the deployment’s discretion and is documented as a project amendment.

15 Conclusion

This paper documented the open standards stack on which the BDF protocol is implemented. The composition principle is layering: each layer addresses a specific concern through a recognised standard, with no layer reinventing what an existing standard already provides.

The seven layers, from sensor to public chain, run: SenML and EPCIS for telemetry and supply chain events; FoodOn and SAREF4AGRI for domain semantics; PROV-O for provenance graphs; W3C Verifiable Credentials 2.0 for the signed envelope; beeson with BMT for canonical encoding; Swarm for content addressed storage with retention economics and DREAM revocation; Ethereum or Gnosis Chain for public anchoring. Together these standards form a complete stack from physical observation to immutable cryptographic commitment.

The vector schema mappings (Section 12) document the field by field correspondence between the BDF abstract attestation vectors and concrete standards based payloads. The worked example (Section 13) walks a single sensor reading through every layer, demonstrating that the stack is implementable and that an integrator can adopt it incrementally.

The versioning policy (Section 14) ensures that BDTs created under the current standard versions remain verifiable indefinitely. Standards revisions affect new BDTs without invalidating historical ones. The protocol’s reference implementation supports parsers for current and historical versions across the stack.

What this paper enables. Engineers building BDF compliant deployments now have a complete reference for implementation: the standards to adopt, the field mappings to follow, the cryptographic suites to select, the anchoring chains to target, the worked examples to use as test fixtures. Standards body assessors evaluating the protocol’s interoperability claims can verify against the published standards documents directly.

What this paper does not do. It does not provide a reference implementation in any specific programming language; that is released as a separate engineering artefact. It does not document the verifier and detector APIs; those are in the protocol’s reference implementation repository. It does not perform performance benchmarking or capacity planning; those are deployment specific and depend on hardware and cohort sizes.

Outlook. The standards stack is expected to evolve in two directions. First, upstream contributions: the protocol is expected to contribute private extensions back to GS1 (CBV freeze drying steps), FoodOn (new cultivar IRIs), and ETSI (SAREF4FOOD extension for food processing). Second, new layer adoption: emerging standards in zero knowledge proof systems, post quantum signature schemes, and decentralised identity may be incorporated as Hardening Pathway extensions when they reach production maturity. Each is a deployment level decision that does not require protocol revision.

The BDF protocol’s commitment is to compose, not to invent. This paper is the catalogue of what it composes and how. Together with the main paper [Fair Food Data, 2026a], the methodology paper [Fair Food Data, 2026b], and the security analysis [Fair Food Data, 2026c], it forms the complete corpus that a deployer, a registry assessor, a peer reviewer, or an implementer needs to engage with the protocol at the appropriate level of depth.

Document History

Versions follow semantic versioning: a major increment changes protocol semantics or normative requirements, a minor increment adds material without breaking existing conformance, a patch corrects errors. The current version is stated on the title page and in the PDF metadata.

Version	Date	Changes
0.1	May 2026	Initial draft, companion document 4 of the BDF corpus.
1.0	24 September 2026	First public release. Version and date fixed, licensed under CC BY 4.0, document history added. No technical changes from the May draft.

References

- C. Bormann. RFC 8798: Additional units for SenML. IETF Standards Track, 2020.
- Damion M. Dooley et al. FoodOn: A harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2(1):23, 2018.
- ETSI. ETSI TS 103 264 V3.1.1: SAREF; the smart applications REference ontology. Online, 2020a.
- ETSI. ETSI TS 103 410-6: SAREF4AGRI: an extension of SAREF for the agriculture and food domain. Online, 2020b.
- Fair Food Data. The Bio-Digital footprint (BDF) protocol: Verifiable records for transformed bioassets via Bio-Digital twins and tiered attestation. 2026a. Companion paper, version 1.0, September 2026. <https://fairfooddata.org>.
- Fair Food Data. BDF methodology for recovered food carbon. 2026b. Companion paper, version 1.0, September 2026. <https://fairfooddata.org>.
- Fair Food Data. BDF security analysis: Threat model, trust assumptions, and formal reductions. 2026c. Companion paper, version 1.0, September 2026. <https://fairfooddata.org>.
- Gnosis. Gnosis Chain: Documentation. Online, 2024. URL <https://docs.gnosischain.com/>.
- GS1. Core business vocabulary (CBV) standard, release 2.0. Online, 2022a.
- GS1. EPCIS standard, release 2.0. Online, 2022b. URL <https://www.gs1.org/standards/epcis>.
- GS1. Global location number (GLN). Online, 2024.
- International Organization for Standardization. ISO 8601:2019 date and time. representations for information interchange, 2019.
- C. Jennings, Z. Shelby, J. Arkko, A. Keranen, and C. Bormann. RFC 8428: Sensor measurement lists (SenML). IETF Standards Track, 2018.

- S. Josefsson and I. Liusvaara. RFC 8032: Edwards-curve digital signature algorithm (EdDSA). IETF Standards Track, 2017.
- NIST. FIPS 180-4 secure hash standard (SHS), 2015.
- Viktor Trón. *The Book of Swarm: Storage and Communication Infrastructure for a Self-Sovereign Digital Society*. Self-published, 2024. ISBN 978-615-01-9983-2.
- Viktor Trón and Henning Diedrich. A dream come true: Deletable content in immutable storage. *ACM (manuscript)*, 2025. Manuscript submitted to ACM, September 2025.
- Viktor Tron et al. Beeson: A canonical object notation for content-addressed data. 2026. Forthcoming.
- W3C. PROV-O: The PROV ontology, W3C recommendation. Online, 2013. URL <https://www.w3.org/TR/prov-o/>.
- W3C. Decentralized identifiers (DID) v1.0, W3C recommendation. Online, 2022. URL <https://www.w3.org/TR/did-core/>.
- W3C. Verifiable credentials data model v2.0, W3C recommendation. Online, 2024a. URL <https://www.w3.org/TR/vc-data-model-2.0/>.
- W3C. Data integrity EdDSA cryptosuites v1.0. Online, 2024b.
- Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Yellow Paper, latest revision online, 2014.