FoodBlock: A Universal Data Primitive for the Food Industry

Whitepaper, Version 1.6 *February 2026 (Revised October 2026)*

ABSTRACT

The global food industry operates across fourteen sectors yet without a shared data standard. Existing standards address narrow domains: barcodes identify products, regulations mandate reporting formats, labelling rules govern packaging. None provides a universal primitive. This paper introduces FoodBlock: a minimal, content-addressable, signed data structure consisting of three fields (type, state, refs) and seven types, in three moods (what is, what must be, what may be), capable of representing both history and intent across any food industry operation. The protocol's core axiom, that a block's identity is the hash of its canonical bytes, makes every record tamper-evident by design. The key technical insight is that reasoning AI systems now make universal protocol adoption viable for the first time. A protocol can be both rigorous enough for a multinational supply chain and accessible to a market stall trader who describes a transaction in plain English. The protocol draws on three converging developments: decentralised data models that eliminated the need for central authorities, AI systems capable of reasoning about structured data and acting on behalf of their operators, and social network architectures that demonstrated how trust and reputation emerge from graphs of interactions between participants. Because the protocol is three fields of valid JSON, reasoning AI systems can read, write, and converse through it natively. An autonomous system does not need a custom integration for every system it encounters. It needs only the protocol. This means a person's autonomous system can act on behalf of a food business within the authority it is granted, negotiating with other autonomous systems, querying businesses directly, verifying certifications with regulatory authorities, and managing inventory, with human approval when necessary. Crucially, these systems do not merely automate existing tasks. They create interactions that would never have occurred: a bakery's autonomous system negotiating prices with a supplier's, a regulator's system contacting a business to verify an expiring certificate, supplier comparisons that would take days to conduct manually. As humans and autonomous systems participate, the network becomes denser and more verified, producing network effects that increase value for every participant. Developers extend this ecosystem by building connected tools: smart fridges that monitor stock levels, POS systems that record every sale, cold-chain sensors that track temperature from warehouse to shelf, all producing the same universal data structure. The protocol is open, federated, and released under the MIT licence, with reference implementations in six languages. This paper presents the protocol's design rationale, the economy of autonomous systems it enables, the developer ecosystem it supports, and its implications for traceability, trust, and economic growth across the entire food system.

1. THE PROBLEM

Every major industry operates on a shared data standard. Banking relies on SWIFT, a single message format enabling any institution to transact with any other. The web relies on HTTP, enabling any client to communicate with any server. Logistics relies on GS1 barcodes. Healthcare relies on HL7. In each case, a common data format allows systems built by different organisations, in different countries, using different technologies, to interoperate.

The food industry is the oldest and largest on earth: three trillion pounds of annual economic activity, fourteen sectors spanning farming to retail, regulation to waste recovery, hundreds of millions of businesses, and billions of daily transactions. It has no equivalent standard. A farmer in Wales records a harvest in a spreadsheet. A mill in Somerset logs production in a separate system. A bakery in London tracks inventory in another. A restaurant orders supplies via messaging applications. A food safety inspector files reports as PDFs. None of these systems interoperate. Food data is scattered across millions of incompatible formats, locked inside platforms that do not connect. It is not preserved in a form that is shared across systems, durable over time, portable across contexts, and usable at the point of decision.

This fragmentation has direct consequences. When a food safety alert is issued, regulators cannot instantly identify every retailer selling the affected product; the process relies on phone calls and emails while contaminated food remains on shelves. Consumers have no means of independently verifying provenance claims such as "organic" or "free-range." Small producers face certification processes designed for large corporations, creating disproportionate barriers to market access. Even for established businesses, the records that do exist remain locked within the systems that produced them. A restaurant's reviews may sit with Google, TripAdvisor, and Instagram. Its hygiene inspections may sit with a council database. Its certifications may sit with the bodies that issued them. Its sales may sit with its own point-of-sale system. The business cannot assemble a coherent account of itself, and neither can a buyer, regulator, or autonomous system acting on its behalf. Community food initiatives (food banks, surplus redistribution, cooperative buying groups) operate in isolation, unable to coordinate because they share no common data layer. The social dimension of food is as fragmented as the commercial one.

Previous attempts at food data standardisation have addressed narrow domains: GS1 barcodes identify products, FDA regulations mandate specific reporting formats, EU labelling rules dictate packaging requirements. GS1 Digital Link extends barcodes into the web, providing URLs for individual products, but solves identification without addressing provenance, trust, or interoperability between arbitrary food operations. GFSI and BRC certifications standardise food safety audits but produce static documents, not linked data. Each solves one problem for one jurisdiction.

The most ambitious attempts have used blockchain. IBM Food Trust built a permissioned blockchain for supply chain traceability, adopted by Walmart for leafy greens tracking after a 2018 E. coli outbreak. Carrefour deployed blockchain provenance for chicken, tomatoes, and dairy. These efforts produced working systems but exposed a fundamental mismatch between the technology and the domain. Blockchain's core innovation is consensus: ensuring that a network of untrusting parties agrees on a single, canonical sequence of transactions without a central authority. This solves the double-spend problem, preventing the same digital token from being transferred twice. Food data does not have this problem. Two restaurants can independently claim to serve the best carbonara, and both records are valid. A kilogram of flour recorded by the mill, the shipper, and the bakery produces three valid observations, not a conflict requiring resolution. No scarcity needs enforcing. No global ordering is required.

What food data does require is authenticity (who created this record and can we verify it?), traceability (where did this product come from and what happened to it?), and interoperability (can systems built by different organisations, in different countries, using different technologies produce and consume compatible data?). All three are achievable with cryptographic signatures, hash-linked provenance chains, and a shared data format, without the computational overhead, throughput limitations, and operational complexity of consensus mechanisms. A market stall trader does not need to sync with a distributed ledger to record a sale. A temperature sensor in a delivery van does not need to pay gas fees to log a reading. A food safety inspector does not need a blockchain wallet to file a report. The barrier to participation must be as low as the data structure is rigorous.

No existing standard, narrow or ambitious, provides a universal primitive: a single, minimal data structure capable of representing any food operation, commercial or social, supply chain or consumer-facing, in any sector, in any jurisdiction.

2. WHY NOW

Two developments converge to make a universal food protocol viable for the first time.

Reasoning AI systems eliminate the adoption barrier that defeated every previous standard. HL7 succeeded in healthcare because hospitals employ IT departments. SWIFT succeeded in banking because banks employ engineers. The food industry has no equivalent technical workforce. Hundreds of millions of food businesses worldwide are operated by people whose expertise is food, not software. Every previous attempt at food data standardisation required participants to write structured data, and the people who produce food data could not. Reasoning AI systems invert this constraint. A protocol simple enough for AI to read and write natively (three fields of JSON, no schema registry, no required SDK) means any person who can describe a transaction in natural language can produce standards-compliant data. The barrier to adoption drops from technical literacy to the ability to speak. This has never been true of any data standard before.

The same property that makes the protocol accessible to humans makes it native to autonomous systems. Because FoodBlock is three fields of JSON, not a complex schema, not a proprietary format, any language model can read, write, query, and reason about it without an SDK or integration layer. This creates a network effect that no previous standard could achieve: every new autonomous system that connects can immediately transact with every existing participant, human or machine. AI companies are already building commercial systems for procurement, scheduling, and customer service. What these systems lack is a shared language for the food domain. The protocol gives them interoperability, and they give the protocol adoption.

3. THE PRIMITIVE

A FoodBlock is a data structure consisting of three fields: **type** (what kind of act the block records), **state** (the block's data), and **refs** (links to other blocks). These three fields are sufficient to represent every food industry operation. Every block is signed by its author, and a block's identity is the hash of its canonical bytes.

The Types

Seven types classify every food operation. Each has a mood, which says what the block claims.

Actor (exists): a participant in the food system: a person, a machine, a circle (a business, household, or community), or a contact.

Thing (exists): anything held, found, or written: a food item, ingredient, tool, place, recipe, share, or schema.

Observe (is): a claim that something is so: a review, inspection, certification, sensor reading, or publication.

Transform (is): a thing changes: harvesting, milling, baking, fermenting, cooking, or composting, in which inputs become outputs; or a thing is kept and stays itself.

Transfer (is): a thing changes owner, holder, or place: a sale, shipment, donation, or booking.

Promise (must): a commitment, offered or asked: an order, an offer, a standing agreement.

Grant (may): authority given to someone: a key, a membership, the right of a system to act for its operator.

These seven types provide complete coverage of the food industry. Whether the context is a fish market, dairy farm, multinational distributor, or fine dining restaurant, all participants create actors, hold things, make observations, perform transforms, execute transfers, make promises, and grant authority. The types are universal; domain-specific details are held in state. Only four types carry authority and are closed, with fields fixed by the specification: actor, promise, transfer, and grant. Every other type, including any new word, is open: a node checks it against the core rules, stores it, and passes it on, but it moves no owner and binds no one. Blocks link by meaning through seven verbs: `updates`, `answers` (the only way to consent), `confirms`, `defeats`, `for`, `derives_from`, and `same_as`.

The moods represent both what has happened and what is planned. A completed shipment is a transfer block; a purchase order is a promise block, offered by one side and answered by the other. An inspection is an observation block; a standing instruction to an autonomous system is a grant that names what it may do. By expressing both history and intent in the same grammar, autonomous systems can reason about future actions using the same structures they use to audit the past. Drafts, beliefs, goals, and budgets are not blocks: they stay on each participant's own node.

Provenance

The third field, refs, is what transforms isolated blocks into a connected graph. Because every block can reference other blocks, a provenance chain forms naturally as food moves through the supply chain. The wheat block names the farm. The flour block derives from the wheat. The bread block derives from the flour. The sale block references the bread. Each step is recorded; each step points to its predecessor.

Each participant records what they know; the chain assembles itself as each participant contributes their segment. In a food safety incident, this graph is traversable: beginning at a contaminated block, following every forward reference identifies every affected product, business, and consumer with precision and immediacy.

Crucially, this traversable graph is not a truth machine, because a signed lie remains a lie. The protocol does not attempt to prevent falsehoods at the point of entry, but instead establishes permanent attribution and preservation. Because every block is cryptographically signed and immutable, any false claim is permanently bound to its author's identity, and a key that signs two blocks after the same predecessor proves its own lie. Trust is therefore not guaranteed by the ledger itself; it is inferred from the consistency of the graph over time, where a single false attestation becomes a permanent, visible liability.

The record fixes what was claimed, by whom, and when. Later records can correct, dispute, interpret, or extend a claim, but they do not silently erase it. Provenance in this system therefore means not only origin, but the recorded sequence of handling, transformation, and transfer through which the good moved. What would otherwise appear as an isolated claim can therefore be situated within a recoverable history.

Content-Addressed Identity

The provenance chain is only valuable if its data cannot be altered after the fact. FoodBlock achieves this through content-addressed identity: when a block is created, its canonical bytes (RFC 8785 with strict writing) are processed through a deterministic hash function, SHA-256 in the default suite, that produces a unique identifier. This identifier becomes the block's permanent identity. Identical bytes always produce an identical identifier, regardless of where or by which implementation the block was written. A reader recomputes the identifier from the bytes and never trusts one it was sent.

Any modification (altering a price, changing a certification date, substituting a supplier name) produces a completely different identifier. Every block in the chain is therefore tamper-evident by design. If any link has been altered, the mismatch is immediately detectable by any party. Because identifiers and signatures require no network connectivity to compute, blocks are valid at the moment of creation. A producer at a market stall without internet access can record a transaction that is complete and verifiable before it reaches any database.

4. TRUST

In the current food system, trust is primarily acquired through purchase: certification badges, platform listings, and audits. These are point-in-time assessments by single organisations, typically renewed on annual cycles regardless of intervening events.

FoodBlock implements a fundamentally different model. Trust is not a credential conferred by an authority; it is computed from the graph of interactions that accumulates as businesses, customers, inspectors, and certifiers create blocks over time. This model draws on the same principle that underpins social networks: reputation emerges from the pattern of interactions between participants, not from a central score assigned by a single entity.

Three layers of evidence contribute to trust computation:

Authenticity. Every block is cryptographically signed by the creating party. This allows any recipient to verify authorship, ensuring that claims cannot be forged or falsely attributed.

Verification depth. A claim made by a single party constitutes a self-declaration. The same claim confirmed (through the `confirms` verb) by an independent customer constitutes peer verification. The same claim certified by a recognised authority (the Soil Association, the Food Standards Agency) constitutes authority verification. A claim can equally be defeated (through the `defeats` verb). Each level carries greater weight due to the increasing difficulty of fabrication.

Consistency over time. A farm that has recorded organic harvests, received positive inspections, and accumulated verified orders over three years possesses a deeper evidence trail than one registered the previous day. Historical depth across multiple independent parties is resistant to manipulation.

No single entity controls trust computation. Different contexts may weight evidence differently: an organic marketplace may prioritise certifications; a wholesale buyer may prioritise verified transaction history; a consumer may prioritise peer reviews. The evidence is consistent; the interpretation is context-dependent. Trust is always current, computed at the moment of query, not frozen in an annual certificate.

5. THE NETWORK

FoodBlock is not an application. It is a layer that applications, devices, and autonomous systems build on top of.

Because the food economy is decentralized institutionally, technically, and semantically, actors require a common syntax through which records can circulate without losing intelligibility. The protocol provides this syntax, serving as an initial grammar rather than a final authority. The primitive is deliberately minimal so that any system can participate. An autonomous system writes blocks in response to a conversation; a legacy database writes blocks via a simple background script; a hardware sensor writes blocks directly. The protocol does not prescribe how data enters, but what it looks like once it arrives. This means adoption is incremental: a single bakery can start by recording its products and sales, deriving immediate value. As suppliers, customers, and inspectors join and contribute their own blocks, the bakery's graph grows richer without any additional effort from the bakery itself. The protocol sits beneath existing workflows rather than displacing them.

Because schemas, vocabularies, and templates are themselves stored as FoodBlocks (a schema is a thing of kind `schema`), the protocol is self-describing, and every closed list the specification uses is published in one machine-readable file, `protocol.json`. An autonomous system connecting to a FoodBlock node for the first time can read the protocol's own field conventions and validation rules directly. It does not need a human to explain the system. Previous food standards require extensive external specification: HL7 requires a 500-page implementation guide, GS1 requires a barcode reference manual. FoodBlock requires three fields. The protocol documents itself using itself.

Each new participant adds value to every existing participant. This is the dynamic that transforms a data format into a network, and it is what creates the surface area for an ecosystem of autonomous systems, connected devices, and developer tools to emerge.

6. AUTONOMOUS SYSTEMS

When autonomous systems can read and write a shared data protocol, they can act on behalf of every participant in the food system, within the authority a grant gives them. Not only businesses, but consumers, regulators, community organisations, and the network itself.

A baker spends the morning making bread, not managing supply chains. A bakery's autonomous system that detects low flour inventory queries the network for suppliers, evaluates offers against the operator's stored preferences, negotiates with suppliers' systems, and presents a draft order for the operator to approve. The operator taps approve and returns to baking. Every step in the exchange that binds anyone is a FoodBlock, signed by its author and referencing the blocks that preceded it; the draft and the operator's preferences stay on the operator's own node. The complete negotiation history is permanent and auditable. Over time the system learns which suppliers deliver on time, which offer the best prices, and which the operator prefers. Routine procurement proceeds in the background while the owner does what they are actually good at. Rather than acting on a static ledger of past events, autonomous systems operate on a graph of active commitments, where offers and negotiations are themselves represented as promise blocks.

But this extends far beyond procurement. Not every system negotiates with another. A delivery-tracking system monitors shipments in real time, verifying that temperature stayed within range and flagging delays before they affect production schedules. A stock-management system watches inventory levels across multiple locations and triggers reorders before shortages occur. A consumer's system discovers local producers, reads verified reviews, and traces the provenance of a product from farm to shelf before recommending a purchase. A regulator's system monitors the network for expired certifications, broken cold chains, and provenance gaps, escalating anomalies to human

inspectors rather than waiting for scheduled audits. A food rescue system detects surplus posted by restaurants and bakeries, matches it to nearby food banks and community kitchens, and coordinates redistribution before waste occurs.

The cumulative effect of these systems is not merely automation. It is amplification. A bakery owner would never spend an afternoon contacting the Food Standards Agency to verify that a new supplier's hygiene certificate is current, or reaching out to five farms to compare organic flour prices, or checking whether a delivery driver's cold-chain records show a temperature breach. But an autonomous system acting by grant does all of this in the background, continuously, without being asked. Each verification, each comparison, each proactive check creates a new block in the graph. The network becomes denser, more trustworthy, and more economically active than any human-only system could sustain. Autonomous systems do not replace human judgment. They increase the number of informed interactions in the system, so that human judgment, when it is applied, operates on better evidence. The graph itself is the reasoning substrate: named edges, typed nodes, and a three-field structure give AI systems a graph they can traverse as naturally as a human auditor follows a paper trail, answering questions like "is this product organic?" not by checking a single field but by following the provenance chain to a signed certification from a recognised authority.

Crucially, when two autonomous systems negotiate, the conversation itself is the record. A bakery system's request for a quote is a promise block, an ask. The supplier's counter-offer is a promise that answers it with new terms. The bakery's acceptance is an answer carrying the status `accepted`, which is the only way to consent. Every message in the negotiation is a FoodBlock, forming a hash-linked chain that constitutes both the communication and the audit trail simultaneously. There is no separate log, no transcript to reconcile, no conversation that happened outside the system. Machine-to-machine communication and human-auditable records are the same thing.

In each case the pattern is the same: an autonomous system reads the graph, applies its operator's preferences, takes action within the boundaries of a grant, and defers to a human for decisions that exceed its authority. The protocol provides the shared language that makes this possible. Without a common data structure, autonomous systems would need custom integrations with every system they interact with. With one, they can transact with any participant in the network from the moment they connect.

7. THE DEVELOPER ECOSYSTEM

A protocol is only as valuable as what people build on top of it. The minimal structure of FoodBlock is designed to enable creativity: not to prescribe what the food industry should look like, but to give developers, businesses, and entrepreneurs the building blocks to shape it themselves.

The barrier to integration is deliberately low. The protocol specifies only the block format, not the hardware, the communication method, or the programming language. A temperature sensor in a walk-in refrigerator writes blocks recording temperature and humidity. A GPS tracker in a delivery vehicle writes blocks continuously, building an unbroken environmental record from warehouse to destination. A POS terminal writes a block for every sale, turning end-of-day reconciliation into a graph query rather than a manual count.

For existing companies, adoption means expanding tools they already sell. A distributor's warehouse management system writes blocks when goods leave the facility. A retailer's receiving system writes blocks when goods arrive. The blocks reference each other across organisational boundaries without requiring shared infrastructure. A POS vendor adds a FoodBlock export to its existing product, and overnight every merchant using that system becomes a participant in the network. The integration cost is low because the protocol asks so little: three fields, valid JSON, a hash, and a signature. Reference implementations exist in six languages (JavaScript, Rust, Python, Go, Swift, and Kotlin), and an MCP server lets any compatible AI system write and read signed blocks.

For new companies, the protocol creates a foundation to build on from day one. A hardware startup designs a smart fridge that tracks stock levels, monitors expiry dates, and triggers reorders through the autonomous systems described in the previous section. A cold-chain startup builds temperature monitors that produce verifiable, tamper-evident records from warehouse to shelf. A mobile app startup builds a consumer product that scans a QR code on any item and renders its full provenance chain, with every certification, inspection, and review attached as verifiable evidence. These companies do not need permission from FoodBlock, do not pay licensing fees, and do not depend on a proprietary API that might change or disappear. They build on an open protocol. A device need not be large: a micro (a sensor, a scale, an oven) needs only the canonical form, SHA-256, Ed25519, its own chain, and the conformance vectors; an edge (the computer at a farm, kitchen, or van) adds local exchange and offline work; a node adds serving others.

The developer ecosystem is not limited to professional software engineers, and this is arguably the most significant consequence of the protocol's minimal design. Previous food data standards failed at adoption because the people who produce food data could not write structured data. Reasoning AI systems eliminate this barrier entirely, and in doing so reveal a deeper property of the protocol: AI is the

universal adapter. The protocol does not need custom integrations with legacy systems because AI is the integration. A photo of a handwritten receipt becomes a FoodBlock. A voice memo describing a delivery becomes a FoodBlock. A PDF invoice from a supplier who has never heard of the protocol becomes a FoodBlock. Every legacy format in the food industry (spreadsheets, paper records, WhatsApp messages, till receipts) becomes compatible the moment an AI can read it and write three fields of JSON. The adoption problem inverts: instead of convincing a million businesses to change their systems, the protocol needs one AI that can read their existing output.

This means the protocol's simplicity is not merely a design preference. It is the mechanism that makes AI-mediated adoption possible. Three fields of valid JSON is a structure that any reasoning system can produce from any input, in any language, without training or fine-tuning. A restaurant owner who has never written code can describe what they need and receive a working tool that reads and writes FoodBlocks. The barrier to participation drops from technical literacy to imagination: anyone who can describe a problem can build a solution.

The result is an ecosystem where hardware manufacturers, software vendors, application developers, and non-technical business owners all produce and consume the same data structure. A sensor built in Shenzhen, a POS system built in London, a mobile app built in Lagos, and a bakery owner's custom inventory tool built through conversation all interoperate without ever coordinating with each other. The protocol is the integration layer.

8. OPEN AND FEDERATED

Everything described in the previous sections depends on one condition: no single entity can own the protocol. A food data standard controlled by one company is not a standard. It is a product, subject to changing terms, increasing prices, and discontinuation. FoodBlock is released under the MIT licence with no patents, no royalties, and no registration requirements. The specification is public (github.com/FoodXDevelopment/FoodBlock). The reference implementations are open source. Any party can implement it: university research groups, government agencies, startups, individual producers, and multinational retailers. A standard must function as a public good.

The deployment model reinforces this principle. The protocol is federated, analogous to email. No single entity controls the network. Any person or machine can run a node, the software that holds their blocks and speaks for them; a business, household, or community is a circle of such participants and keeps its record on its members' nodes. A farm in Pembrokeshire, a mill in Somerset, and a bakery in Bermondsey each retain ownership of their data, on their own infrastructure, under their own control, while maintaining the ability to discover peers, reference each other's blocks, and construct provenance chains that span organisational boundaries. Because a block's identity is derived from its content, the same block stored on multiple nodes is provably identical. If one node goes offline, a copy on another is equally valid. Peers exchange blocks with two messages, pull and push, and every answer a node gives is signed.

Discovery follows the same federated model. A node publishes a well-known endpoint (`.well-known/foodblock`) that declares who the node is, what it holds, who it speaks for, and its admission rule. Nodes are found the way email addresses are: given directly, listed in any directory, or followed from a ref, so following a provenance chain from a bakery's bread to a mill's flour implicitly discovers the mill's node. Nearby, a node may announce itself on a local network. No mechanism is mandatory. A node with no peers is still valid. The protocol provides the format; the network assembles itself through use.

This federated architecture matters because the alternative is platform dependency. When a business builds operations on a proprietary platform, it does not own its data. If the platform modifies its terms, increases prices, or ceases operation, the business loses access to its own history. Federation eliminates these single points of control. Every participant owns their data, chooses their infrastructure, and interoperates through the protocol alone. This shift is not merely technical, but economic: it turns data into participant capital rather than platform capital. In a platform-dominated economy, asymmetric legibility prevails. The central platform acquires practical power over how a business is seen and reached, while the business itself cannot easily move, combine, or reuse its record of activity. By making data portable, the protocol restores exit rights as a condition of bargaining power. Intermediaries must compete to serve the participant, rather than owning the history that makes the relationship valuable.

The protocol is small, and its extension is open. Any party may write a schema, and any new type word is an open type that every node stores and passes on, while authority stays with four closed types. The specification is versioned as a whole; FoodBlock 1.6 is the first version of the protocol itself, and data from earlier systems enters once, through a converter that is not part of the protocol. Contributing to the protocol requires the same capabilities as using it.

9. TECHNICAL PROPERTIES

The previous sections describe what the protocol enables. This section describes the technical properties that make it work. These are not implementation details buried in source code. They are architectural decisions that directly affect the reliability, security, and practicality of every operation described in this paper.

Immutability and append-only history. A FoodBlock cannot be edited once created. Its identity is its content, so modifying any field produces a different identity entirely. Updates are expressed as new blocks that reference their predecessors, preserving the full history of every change. A product's price history, a supplier's evolving certifications, and a system's commitments are all preserved as chains of blocks. Nothing is overwritten. Nothing is lost. Each signing key also has its own chain: every block names the key's previous block (`prev`) and carries the next sequence number (`seq`), so two blocks of one key naming the same predecessor prove a lie. This property is what makes the provenance chains described in Section 3 trustworthy: if any block in a chain had been silently altered, the hash mismatch would be immediately detectable by any party.

Offline validity. Hashing and signing require no server, no network connection, and no authority. A block's identity can be computed anywhere, by anyone, using only the canonical bytes of the block and the SHA-256 algorithm. This means a farmer at a rural market without mobile signal, a fisherman at sea, or a delivery driver in a tunnel can all create valid, signed blocks that are complete and verifiable before they ever reach a database. A device with no clock may leave out the time, and the edge that receives its block records when it arrived. When connectivity returns, blocks sync to the network and deduplicate automatically: if the same block was created independently by two systems, the identical hash ensures only one copy exists.

The same data always produces the same identity. For content-addressed identity to work across different systems, languages, and platforms, the same block must always produce the same hash. FoodBlock achieves this through a canonical form: RFC 8785 JSON with strict writing. Keys are sorted, text is Unicode-normalised (NFC), times are exact UTC to the millisecond, amounts with a fraction are decimal strings, money is `{ minor, currency }`, sets of references are written sorted, and `null` and floating-point numbers are refused. A node never repairs a block: it refuses it, and the author signs a new one. Reference implementations exist in JavaScript, Rust, Python, Go, Swift, and Kotlin. Rust holds the one core of every rule, and the other languages are thin layers over it. Conformance vectors verify that every implementation produces identical hashes for identical inputs, and a second implementation written from the specification text alone checks the specification itself. If they ever disagree, the protocol is broken.

Every claim has a verifiable author. Every block is signed by its author. The default suite uses SHA-256 and Ed25519 digital signatures; `p256` (ECDSA, including passkeys) and `sm` (SM2 and SM3) are two further suites. A message wraps the block with its signature and the protocol version. Any recipient can verify that a block genuinely originated from the claimed author without contacting any central authority. An identity begins with a genesis block signed by its own key, and later keys, devices, and recovery are added by grants. Signing is what transforms self-declared data into verifiable claims: a certification signed by the Soil Association carries weight because the signature is cryptographically bound to the Soil Association's public key.

Businesses control who sees what. Not all data should be public. A supplier's cost price, a business's margin, a recipe's formula may need to remain confidential. Every block carries an audience: either public, or a sorted list of commitments, one per recipient, so that recipients do not see each other. A commitment is a salted hash that hides a value until the salt is shown. A private block's body is sealed in an envelope encrypted to its audience (AES-256-GCM under a content key, agreed with each recipient by the suite's key agreement), while its public header, consisting of author, sequence number, previous block, and hash, stays readable. This allows any party to check a chain, its gaps, and its forks, and only the audience reads the bodies. The protocol is candid about the cost: the header shows how many blocks an author wrote, and when. A block cannot be unsent, but a node stops serving older versions once a newer one narrows the audience. Anyone a block is about is in its audience, and in a business's blocks a customer is a commitment, so a worker's chain shows how many customers, never who.

Progressive disclosure. Rather than relying on complex cryptographic proofs to selectively disclose individual fields of a single block, the protocol achieves privacy and verification through the structure of the graph. Because the data model is a graph of distinct, signed blocks (separating products, harvests, certifications, and transfers), a business can share high-level public blocks (such as a product listing or a signed organic certification) while keeping sensitive blocks (such as cost prices or supplier identities) sealed. When a counterparty requires additional verification, the owner, or the owner's autonomous system acting by grant, can send the specific supporting blocks. This keeps the base protocol simple while allowing trust to be verified dynamically.

Continuous data streams are represented efficiently. Temperature sensors, production line monitors, and delivery tracking systems generate data continuously. Rather than creating a block for every reading, a reading carries its amount, accuracy, and coverage, and names the sensor it came from; a whole stream is linked by its Merkle root, which proves any single chunk without sending the rest. Raw streams live in time-series databases; FoodBlocks capture the meaningful moments. A sensor reading is evidence that a confirmer may cite and never settles a

promise alone. This keeps the network efficient while maintaining a tamper-evident audit trail.

Data can be erased without breaking the chain. The append-only model preserves auditability, but legal requirements such as GDPR may require erasure. The optional erasure profile resolves this tension. Personal fields (names, phone, email, address, and similar) are never written as plain values in a block about a person; each is a commitment. An `observe` of kind `erase` asks nodes to delete the salt and value they hold. The block still verifies, its hash, type, and references are preserved so the graph stays intact, and without the salt nobody can link the hash to the person. The profile is the mechanism and not legal advice: the hash of a personal value may still count as personal data in some jurisdictions.

Structure is checked where authority is at stake. Open types accept any valid JSON in state, which keeps the primitive universal. The four closed types (actor, promise, transfer, grant) have fixed fields, and a block carrying an undeclared field is refused unless a schema it names declares it. Schemas are themselves stored as FoodBlocks: a schema states the type it targets, its fields, the ref roles it adds, the grounds (the real event without which a block of that kind is a lie), and labels in each language. Words people read live in schemas, never in the protocol.

Conflict resolution. The protocol detects conflict through each key's chain. A block is refused if it does not fit its key's chain, and two blocks of one key naming the same predecessor prove a lie: both are kept and neither binds anyone else. A record that must be ordered, such as an offer being allocated, has one orderer chosen by whoever opens it, and its binding blocks form one chain. Consent decides what binds, not what exists, so the approval of another party is itself a block in the graph, making every transfer of control visible and auditable.

Compact transport. For links where JSON is too heavy, such as a radio, a tag, or a sensor, a public block may travel as CBOR (RFC 8949). This is a transport form only: the identifier is always the hash of the canonical JSON, and a receiver rebuilds the block, hashes it, and verifies the signature over that JSON. The integrity layer is never compromised.

No consensus mechanism required. As established in Section 1, food data requires authenticity, traceability, and interoperability, not scarcity enforcement or global transaction ordering. FoodBlock adopts the useful properties of distributed ledgers and discards the consensus mechanism entirely.

10. IMPLICATIONS

None of the capabilities described in this paper exist in isolation. Provenance is only useful if the data is tamper-evident. Trust is only computable if the graph is open and federated. Autonomous systems are only effective if they share a common language. The developer ecosystem is only viable if the protocol is minimal enough for anyone to implement. Each section of this paper depends on every other. The implication is that the food industry does not need better tools in any single category. It needs the layer beneath the tools.

When that layer exists, the economic consequences are significant. Every hour a business owner currently spends chasing invoices, comparing suppliers, or manually reconciling inventory is an hour not spent on the work that actually generates value. A shared data protocol, combined with autonomous systems that handle routine operations, moves humans higher in the decision-making process. The baker decides what to bake, not which supplier to call. The farmer decides what to grow, not how to format a compliance report. The inspector decides where risk lies, not how to assemble documentation. Human attention shifts from operational mechanics to judgment, creativity, and strategy. The economic activity of the food industry does not shrink when routine tasks are automated. It grows, because the people who understand food best are freed to do more of what they are good at.

The food industry's data fragmentation is not a technology deficit. Databases, platforms, and applications are abundant. It is a primitives problem: the fundamental data structure that allows incompatible systems to produce compatible data does not yet exist. FoodBlock provides that primitive. Three fields, seven types, one axiom. The structure is minimal enough for a market stall trader to use through conversation with an AI assistant, and rigorous enough for a multinational supply chain to build automated operations upon. The protocol is open, federated, and designed to be owned by no one and used by everyone.

APPENDIX: SECTOR COVERAGE

The food industry comprises fourteen interconnected sectors. The following reference demonstrates how seven types provide complete coverage without requiring any sector to modify its operations.

- **Primary Production:** farms, fisheries, dairies. Actors produce things through transforms. A dairy farm (actor) at a location (thing) produces milk (thing) through milking (transform), recorded with organic certification (observe).
- **Processing and Manufacturing:** mills, factories, abattoirs. Raw things are transformed into processed things. A mill (actor) grinds wheat (thing) into flour (thing) through milling (transform), tracked by batch inspections (observe).
- **Distribution and Logistics:** warehouses, shipping, cold chains. Things are transferred between places. A shipment (transfer) moves flour from a mill

(thing) to a bakery (thing), agreed by a promise and monitored by temperature readings (observe).

- **Retail:** shops, markets, online stores. Actors offer things (promise) and sell them to consumers through transfers.
- **Hospitality:** restaurants, cafes, hotels. Actors transform things into meals and serve them at places, reviewed by customers (observe).
- **Food Service:** catering, institutional kitchens, meal planning. Transforms of things at scale, recorded as observations.
- **Waste and Sustainability:** food rescue, composting, redistribution. Surplus things are transferred to new actors through donations (transfer).
- **Regulation and Food Safety:** inspectors, certifying bodies, standards agencies. Authority actors observe other actors through certifications and inspections.
- **Education and Media:** food writers, educators, content creators. Actors create observations about things and places.
- **Community and Social Food:** food cooperatives, community kitchens, social enterprises. Circles of actors share things through transfers, admit members by grant, and record community events as observations.
- **Health and Nutrition:** dietitians, nutritional analysis, health claims. Professional actors observe things with assessments and dietary plans.
- **Food Finance:** commodity trading, agricultural investment, insurance. Value is transferred between actors, observed with market data.
- **Cultural Food:** heritage foods, geographical indications, traditional methods. Things and transforms are observed with cultural certifications tied to places and regions.
- **Food Technology:** novel ingredients, novel processes, food science research. Innovator actors experiment with transforms of things, observed through publications and experimental data.

No sector requires a type outside the seven. The three-field structure adapts to every context through open kinds and domain-specific state properties.

FoodBlock is an open protocol released under the MIT licence. This document, like the protocol it describes, is available for unrestricted public use. The specification and reference implementations are at github.com/FoodXDevelopment/FoodBlock, and this paper is at foodx.network.