FoodBlock: An Open Language for the Food Economy

Whitepaper, Version 1.9 *November 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. A signed block binds only its signer: a promise pledges what the signer owns, never the signer themselves. No one governs the protocol, and it fills in no one's silence: nothing binds anyone that they did not sign. Her AI acts only under her grant, the worst she can lose is her own figures, and a deal's disputes go only to a judge both sides chose. A signed block keeps the meaning it had when signed, and the rules are checked by two engines built apart. 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. It draws on decentralised data models, which removed the need for a central authority; on social networks, which showed that reputation emerges from graphs of interaction; and on AI systems that can reason about structured data and act for people under their grants. Because the protocol is three fields of valid JSON, reasoning AI systems can read, draft, 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 the person's approval wherever their grant does not already cover the act. 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. The protocol is open, federated, and released into the public domain (CC0), with two independent engines and SDKs in six languages. This paper presents the protocol's design rationale, the economy of autonomous systems it enables, the developer ecosystem it supports, its limits, and its implications for traceability, trust, and voluntary exchange across the food economy. Its purpose is to increase economic freedom in the world's food economy.

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: 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, no one can quickly reach everyone holding the affected food; 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 (Walmart, 2018). Carrefour deployed blockchain provenance for chicken, tomatoes, and dairy (CIO, 2018). 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. Where scarcity does matter, as when a lot is sold, the owner's own signed chain orders the sales so a second draw on the same lot takes nothing, while two conflicting chains signed by the same key prove a lie once both are seen. 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, gives strangers in the food economy a way to bind themselves without a middleman—a single, minimal primitive built from the signed word, capable of representing any food operation in any sector and 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 and no schema registry) means any person who can describe a transaction in natural language can produce standards-compliant data. The barrier to producing structured records drops from technical literacy to describing a trade in plain language.

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, draft, query, and reason about it without an integration layer; a small library then hashes and signs what the person approves. Every new autonomous system that speaks the protocol can immediately exchange blocks with any willing participant, human or machine, within its principal's grant and the receiver's admission rule. 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.

This makes one question urgent: when an AI agent trades for a person, who is bound? A machine owns nothing and cannot be held to a promise, because only persons act (Mises, 1949). Existing messaging and API protocols cannot tell a supplier, from the record alone, whether a person consented to an agent's order, what limit she set, or which of her goods or funds stand behind it. Without a grammar that traces every machine action back to a person's signed grant and stated figures, autonomous trade either stalls on manual approval for every step or exposes people to unbounded commitments they never signed.

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 one of three moods—**is** (a reading), **must** (a promise), or **may** (a grant)—which says what the block claims.

Actor (is): a participant in the food system: a person, an autonomous system, a machine, a circle (people who own things together in parts, such as a business, household, or community; a circle never signs and holds no key), or a contact (someone recorded by another, who holds no key of their own).

Thing (is): anything held, found, or written: a food item, ingredient, tool, place, recipe, share in a business, 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, as when cheese ages in a store or grain is held in a silo.

Transfer (must): a promise whose condition is already met, in which a thing changes owner, holder, or place: a sale, shipment, donation, or booking. Storing moves only the holder (`to_holder`) and the goods stay the depositor's; lending interchangeable goods moves the owner (`to_owner`) together with a promise of the same kind back (Huerta de Soto, 2006).

Promise (must): a commitment, offered or asked: an order, an offer, a standing agreement, or an undertaking not to do something. Every promise has one form: if a stated condition holds, something of the signer's becomes the other side's. A judge, if the sides name one, settles a quarrel; a promise with none binds all the same.

Grant (may): the signer's consent to who may speak as her (a key grant) or act for her or on what is hers within stated limits—such as letting an autonomous system act under her promise, or passing on a narrower part of a grant she holds.

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 and checked only against core rules; whether a node stores or forwards it is that node's own promise, and an unknown open type gives no authority, moves no owner, and binds no one. Blocks link by meaning through seven verbs: `updates`, `answers` (the only way to take up an offer or an ask), `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.

What a signature binds. Only a signed word binds, and it binds only its signer, never their body. It binds in only three ways: the signer's name to the truth of what it records (is), the signer's property to the condition it names (must), and the signer's consent to another acting within stated limits (may). Nothing fills anyone's silence: silence counts only by the silent person's own earlier signed word about what her silence will mean, and a signer cannot deny her own signed word (Kinsella, 2003). Title moves only by the previous owner's signed word, or by first use of the unowned (Rothbard, 1982; Hoppe, 1993). A side may always stop a promise from that point onward; what is then owed is only what the promise pledged, and the record shows what each side did. This is the title-transfer theory of contract (Evers, 1977; Rothbard, 1982; Kinsella, 2003): a promise is a conditional transfer of title, *if this happens, this of mine becomes yours, and this person settles it*. Trade, credit, wages, deposits, warranties, guarantees and stakes behind a claim are all instances of this single form, and none requires a type of its own. A promise to pay later carries the condition that the signer holds title to the sum then, and no block reaches unpledged property or a person's body, which was never acquired and cannot be handed on (Kinsella, 2003): debt bondage cannot be written in the protocol.

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 and following every block that references it, across the nodes permitted to read them, identifies the affected products and businesses, and each business can reach the customers it recorded.

This traversable graph is not, however, 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 (strict UTF-8-ordered JSON, matching RFC 8785 up to U+FFFF) are processed through a deterministic hash function, SHA-256 in the `std` 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.

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, and it is not a score the protocol computes. The protocol records facts, each signed by the party who states it; whether someone can be relied on is each reader's own judgement from those facts. Reputation emerges from a participant's record of voluntary exchanges as each counterparty weighs it—a dynamic social networks later illustrated at scale—not from a central score assigned by a single entity. Nobody owns a reputation: it is what others think (Rothbard, 1982).

Three kinds of evidence are open to every reader:

Authenticity. Every block is cryptographically signed by the creating party. This lets any recipient verify which key signed those bytes, so a claim cannot be attributed to a key that did not sign it.

Independent confirmation. A claim made by a single party is a self-declaration. Others may confirm it (through the `confirms` verb) or defeat it (through `defeats`): a customer who received the goods, an inspector, or a certifying body such as the Soil Association. `confirms` means its signer signs that block's word too: where she holds a part, a voice, or a place in it, she joins it; where nothing in it is hers, it is her word that it is true and binds nothing. A claim's schema names who is in a position to confirm it, and a reader may weigh a confirmation by how hard it would be to fabricate. The protocol fixes no order of weight.

Consistency over time. A farm that has recorded organic harvests, received inspections, and accumulated orders over three years has a deeper evidence trail than one registered the previous day, and a history spread across many independent parties is hard to manipulate. Part of this evidence is the record of promises: for each one, the protocol states who answered it, how and when it ended, what moved and what was paid, and when it was due. It never states whether a promise was kept. That verdict belongs to the reader.

Disputes. Nobody rules by default, and the law is not in the protocol. Disputes enter through two doors onto one record. A dispute over a deal—including fraud in that deal, which is a condition that failed (Rothbard, 1982; Kinsella, 2003)—goes only to the judge, or panel of judges, that both sides chose, named in the deal's schema grounds or by a later message both signed; a side never judges its own deal. An attack outside any deal—theft, poisoning, drift, fraud harming an outsider, or a word signed under threat—has no agreed judge and goes to the victim's own judge, whose jurisdiction does not ask the attacker's consent (Rothbard, 1982; Kinsella, 2003). In both doors, a ruling is a signed fact, never an order: in a deal it changes which blocks count or releases a bond whose condition names that judge; in an attack outside any deal it moves nothing in any fold, since title moves in the record only by the previous owner's signed word. The judge answers for a ruling if it was wrong, and whoever acts on it answers for the act.

No single entity controls trust. Different readers weigh the same facts differently: an organic marketplace may weigh certifications most, a wholesale buyer its counterparties' payment history, a consumer peer reviews. The facts are the same for every reader; the judgement is each reader's own, made at the moment it is needed rather than frozen in an annual certificate. Value and risk are subjective (Menger, 1871), and whether a given counterparty will honour a single future delivery is a one-off case with no mathematical probability (Mises, 1949), so no protocol can compute a score in place of the person who bears the loss.

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 decentralised 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 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 node for the first time can read those schemas and closed lists directly without a custom integration guide. Previous food standards require extensive external specification: HL7 relies on lengthy implementation guides, GS1 on its barcode reference manuals. FoodBlock's rules are published in the protocol itself. The protocol documents itself using itself.

Identity requires no central service or registry. A person's identity begins with a genesis block signed by a key made on a device she holds, and who may speak as her is her own signed list of voices—signing keys, hidden key hashes, or named people—with no voice, count, wait, or clock filled in for her. A stolen key is bounded by her own figures: a voice may carry per-act (`each`) and cumulative (`limit`, optionally per period `every`) bounds, together with a `reach` delay on clocks she names before its moves stand, so a higher voice or her named people can freeze or cut off a stolen key before it spends past the figures she set.

Each willing participant widens the set of counterparties with whom others can verify records and trade without an intermediary.

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, and community organisations.

A baker spends the morning making bread, not managing supply chains. Her bakery's autonomous system detects low flour inventory, queries the network for suppliers, evaluates offers against her stored preferences, negotiates with suppliers' systems, and presents a draft order for her to approve. She 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 her preferences stay on her own node. The complete negotiation history is permanent and auditable. Over time the system learns which suppliers deliver on time, from the record of each supplier's past promises, which offer the best prices, and which she prefers. Routine procurement proceeds in the background while she does what she is 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 suggesting a purchase. A regulator's system reads what is public and what holders have chosen to grant it (a recall grant, for example), and flags expired certifications, broken cold chains, and provenance gaps 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, within its grant and the thinking she pays for, without being asked each time. Each verification, each comparison, each proactive check creates a new block in the graph. Each participant gains a deeper, independently verifiable record of counterparties and prices than manual checking 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 links, typed blocks, 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 an authority the reader trusts.

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 bind an offer. 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 is a tool with its own key, signing under her grant; it never signs as her, because only persons act (Mises, 1949). What binds her is her own signed grant and the promises it names, never the machine's bare word. Every block under a grant is metered along one chain (`refs.meter`), and from her grant and its named promises alone, plain code computes the worst case—the most the grant can bind her to hand over (`at_most`) and the most she can lose (`can_lose`)—bounded strictly by her own figures. A grant may limit both the total a system commits and the size of any single act, in money or in quantities of goods, and a block signed as a circle binds its owners' common things only when enough of its owners' parts, or of the hands a row names, sign for it. 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 FoodBlock signing to its product, and each merchant who creates a genesis key can immediately sign and exchange records in the shared format. The integration cost is low because the protocol asks so little: three fields, valid JSON, a hash, and a signature. Six SDKs (JavaScript, Rust, Python, Go, Swift, and Kotlin) sit over one Rust core engine, checked against a second independent engine written from the specification alone, alongside an MCP server, a CLI, and a reference host so any compatible AI system can read blocks and draft new ones for its user to sign.

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; a dealer handles a single deal under eight rules without running a server; 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. As Section 2 argued, reasoning AI systems remove the barrier that defeated earlier standards, and here that has a further consequence: AI is the universal adapter. A photo of a handwritten receipt, a voice memo describing a delivery, or a PDF invoice from a supplier who has never heard of the protocol can each become a FoodBlock, so the protocol needs no custom integration with legacy systems. A restaurant owner who has never written code can describe a tool and receive one that reads and writes FoodBlocks.

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 into the public domain (CC0), with no patents, no royalties, and no registration requirements: an idea is not property (Kinsella, 2008), so nobody, its authors included, holds a claim on it. The specification is public (github.com/FoodXDevelopment/FoodBlock). The reference implementations are in the public domain too. Any party can implement it: university research groups, government agencies, startups, individual producers, and multinational retailers. A standard must be owned by no one: anyone may use it without asking.

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 owners' nodes. A farm in Pembrokeshire, a mill in Somerset, and a bakery in Bermondsey each hold their own record, on machines they own, 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, which rule sets it runs (`reads`), its admission rule, its endpoints, and its public log when it keeps one. 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 hold its own record. 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 holds their own record, 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. Brokers who carry, store, or lend remain entrepreneurs competing to serve the participant, rather than privileged intermediaries who own 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 any node can validate against core rules without knowing its semantics—storing or forwarding it only by its own rule—while authority stays with four closed types. Lasting meaning is preserved through rule sets named by hash: a rule set is the SHA-256 hash of its manifest (`spec/rule-set.json`), which hashes the specification files and test vectors, while a number such as "1.9" is only a label for people. A signed block names its rule set in `state.rule_set` (an absent field names the first locked set, H_0) and keeps the exact meaning it had when signed—its bytes, its id, whom it speaks for, what it binds, and who owns what it moves—even as later rule sets improve the protocol. Each reader runs the rule sets it chooses; an old reader that meets a block naming a rule set it does not run holds it as `unknown_rules` and answers less, never otherwise. 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.

No one governs FoodBlock. It is in the public domain, so anyone may publish a version of it, and each node chooses which version it speaks; the version FoodX's people maintain is one among any others. The version FoodX's people maintain is fixed once nodes that FoodX does not run, and an implementation FoodX did not write, pass its conformance vectors.

The protocol also fills in no one's silence. It holds no defaults: a deal states its own terms, a group states its own rules, and a node states its own admission rule. Where a word is left out, the protocol reads nothing into it, so no rule binds anyone that they did not sign.

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.

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 and deduplicate automatically: if the same signed block arrives through two different routes, the identical hash identifies it as one copy.

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 strict canonical JSON form that sorts object keys by UTF-8 byte order (matching RFC 8785 for every key up to U+FFFF). Keys are sorted, text is well-formed UTF-8 compared byte for byte with no normal form and no tables in the protocol, times are exact UTC to the millisecond, amounts are decimal strings written as `{ value, unit }` (with an optional `of` naming the issuer's kind, since money is simply the most saleable good and a claim is not the good itself; Menger, 1892; Mises, 1953), 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. Six SDKs 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. Two engines built apart—the Rust core and an independent JavaScript oracle written from the specification alone—agree on every conformance vector (3,007 door cases and 2,224 frozen audit cases). A necessity table maps every protocol feature and all 27 invariants to tests so no rule is untested or dead, and differential fuzzing runs thousands of random-world and property checks across both engines. If the two engines ever disagree, the protocol is broken.

Every claim has a verifiable author. Every block is signed by its author. The `std` suite—the fixed spelling of an omitted suite field, an encoding rather than a default—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. 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, or sealed with an `xwing:` post-quantum hybrid key combining ML-KEM-768 and X25519, while a post-quantum signing suite is reserved as a future row within the 8,192-byte signature limit), while its public header, consisting of author, sequence number, previous block, and hash (plus `key` and `skip` when present), 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 a key's sequence number (unless blurred with `skip`), and a carrier sees when blocks are sent and how large they are. A block cannot be unsent, but a node that honours its author may stop serving older versions once a newer one narrows the audience, and a business may write a customer as 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, yet a node may promise the people it serves that it will forget them (some nodes promise this to follow laws such as GDPR). The optional erasure profile makes that promise keepable. A node that has made this promise writes personal fields (names, phone, email, address, and similar) only as commitments. 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: only nodes that promised to erase delete the salt, any counterparty who already received the opening still holds it, and 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 valid but held not read—kept as its header and counted in no fold—unless a schema it names declares it. Schemas are themselves stored as FoodBlocks: a schema states the type it targets, its fields, the ref keys 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 whose `prev` or `seq` does not fit its key's chain is still valid—validity never reads `prev`—but takes nothing in any fold, and two blocks of one key naming the same predecessor or sequence number form a pair that proves equivocation: both are kept, neither binds anyone else, and neither frees their signer. An offer that leaves a side open is allocated only by its offerer's own signed yes, drawn along her chain; where order or time across parties matters, the sides name a clock whose chain of signed receipts orders it. 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.

Commitments follow real time and end cleanly. A standing promise repeats by a calendar rule in a named time zone, under any calendar that rule supports, such as the Hijri or Chinese lunar calendar, or a set interval after an event, such as one week after each delivery. A promise may depend on a condition that any node can check from the graph: a debt settled, a deed recorded, a reading within range, or a signal from enough named sources. A condition decides whether, and when, a promise is owed; it never settles one. A key grant that leaves no voice (`voices: []`) closes an identity for good, ending every standing word the person gave in her own name (grants, offers, requests, transfers waiting on an answer, account words, and hands); the record says closed, never why. A will is her own transfer conditioned on `{ closed: <her id> }`: in life it holds nothing and she may end it, and at the close her wills take in the order of her chain, after every other draw of hers. A close ends no binding promise: what was already accepted stands as signed. A stop names no reason; a side who wants to say why does so in its own words, in a separate note on the deal, which informs people and changes nothing the protocol computes.

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 global transaction ordering. Where scarcity matters, the holder of the scarce thing orders its own record (Conflict resolution, above). FoodBlock adopts the useful properties of distributed ledgers and discards the consensus mechanism entirely.

10. LIMITS

A protocol with no central authority and no global consensus trades global prevention for local proof. FoodBlock is explicit about what it does not keep.

Double sales and unseen chains. Because FoodBlock runs offline with no global ledger, a dishonest seller can sign two sales of the same lot to two offline buyers. Once the chains meet, two sales on forked branches of the same key prove the lie, and neither buyer is shown as owner while the seller stays bound to both (if sequenced on one chain, the first takes and the second takes nothing); but if two buyers never compare records or never read the seller's chain whole back to its start, neither sees the other's sale.

A stolen key. The record cannot tell a taken key from its owner's hand. Until she is warned and a higher voice or her named people freeze or cut off the key, a thief holding her only key can sign blocks that look like hers. Bounds (`each`, `limit`, `every`) and `reach` delays cap what she can lose, but only where she configured them in advance.

What hosts, AI providers, and carriers see. Sealing hides block bodies from outsiders, not from the machine that runs a person's AI: whoever runs her AI sees everything it reads. A carrier or host that routes sealed copies sees headers, sizes, timing, and which mailbox or reader a push names; a

single host serving both sides sees who deals with whom, whatever is sealed. Sealing also gives no forward secrecy (compromise of a decryption key opens past envelopes) and no deniability: once shown, a signed block proves its author forever to anyone who holds it.

Free keys and Sybil identities. Because identity requires no central registry, generating keys costs nothing and one person can look like many. Reputation and admission rules cannot rely on raw counts of unknown keys; readers must weigh history, independent confirmations, and parties they already know.

Unbounded offers. An offer with no `expires` and no named clock cannot be withdrawn from someone who accepts it unseen. Whoever wants an offer to end must write an end into it.

The record is not the world. A signature says what its signer binds, never that she can pay or holds the stock in her warehouse. FoodBlock records claims, transfers, and rulings as facts; enforcement lies outside the protocol.

Field testing. While the core engines, SDKs, and SMS read-back renderers pass automated suites, nothing has yet been tested in the field on a basic phone.

11. IMPLICATIONS

None of the capabilities described in this paper exist in isolation. Provenance is only useful if the data is tamper-evident. Each reader can judge trust only 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 people trading in food do not merely lack better software tools; they lack a way for strangers to bind themselves without a middleman, built from the signed word.

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. Automating routine checks does not replace human judgement; it leaves producers and buyers freer to act on real prices from real exchanges—the basis of economic calculation (Mises, 1949)—and to trade on terms they choose.

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 autonomous system, and rigorous enough for a multinational supply chain to build automated operations upon. The protocol is open, federated, and in the public domain.

The deeper implication is economic. Because a signed block binds only its signer, and binds what they own rather than who they are, the same small set of rules applies to a smallholder and a multinational, and no participant can be held in place except by what they themselves pledged. Records that participants hold, and promises that bind only property, are what allow more people to decide, own, and trade on terms they can refuse. That is the purpose of the protocol: to increase economic freedom in the world's food economy.

REFERENCES

Bormann, C., & Hoffman, P. (2020). *Concise Binary Object Representation (CBOR)*. RFC 8949.

CIO (2018). Carrefour modernizes food traceability with blockchain. https://www.cio.com/article/3323073/carrefour-modernizes-food-traceability-with-blockchain.html

Evers, W. M. (1977). Toward a Reformulation of the Law of Contracts. *Journal of Libertarian Studies*, 1(1), 3–13.

Hoppe, H.-H. (1993). *The Economics and Ethics of Private Property*. Boston: Kluwer Academic Publishers.

Huerta de Soto, J. (2006). *Money, Bank Credit, and Economic Cycles*, ch. 1. Auburn, AL: Ludwig von Mises Institute.

Josefsson, S., & Liusvaara, I. (2017). *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032.

Kinsella, N. S. (2003). A Libertarian Theory of Contract: Title Transfer, Binding Promises, and Inalienability. *Journal of Libertarian Studies*, 17(2), 11–37.

Kinsella, N. S. (2008). *Against Intellectual Property*. Auburn, AL: Ludwig von Mises Institute.

Menger, C. (1871). *Principles of Economics*. Trans. J. Dingwall and B. F. Hoselitz (1976). Auburn, AL: Ludwig von Mises Institute.

Menger, C. (1892). On the Origin of Money. *The Economic Journal*, 2(6), 239–255.

Mises, L. von (1949). *Human Action: A Treatise on Economics*. New Haven, CT: Yale University Press.

Mises, L. von (1953). *The Theory of Money and Credit*. New Haven, CT: Yale University Press. (Originally published 1912.)

Rothbard, M. N. (1982). *The Ethics of Liberty*, ch. 9, 10, 12–13, 19. Atlantic Highlands, NJ: Humanities Press.

Rundgren, A., Jordan, B., & Erdtman, S. (2020). *JSON Canonicalization Scheme (JCS)*. RFC 8785.

Walmart (2018, September 24). In wake of romaine E. coli scare, Walmart deploys blockchain to track leafy greens. https://corporate.walmart.com/news/2018/09/24/in-wake-of-romaine-e-coli-scare-walmart-deploys-blockchain-to-track-leafy-greens

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 the mill's warehouse (thing) to the bakery's kitchen (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. Things are transformed at scale, with temperature and allergen checks 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. They sign observations (certifications and inspections) like any other actor; the protocol grants them no special authority, and state mandates remain force outside the protocol.
- **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 hold parts in common, share things through transfers, set rows or grants for who may act, 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. Loans, forward contracts, and insurance are promises; payments are transfers; market prices are observations.
- **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 into the public domain (CC0). 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.