The Open Cap Table Protocol

A specification for the onchain register of record

The onchain extension of the Open Cap Table Format. Every issuance, transfer, conversion, exercise and vesting event is recorded onchain as the issuer's official register: the book itself, not a copy of one. One protocol, one data model, for private and listed issuers alike.

AUTHORS

Joris Delanoue, Fairmint

Thibauld Favre, Fairmint

Contents

Abstract 1

1. Introduction 3

2. OCP and the Open Cap Table Format 6

3. Design Principles and Register Semantics 7

4. System Overview 13

5. Core Data Model 15

6. Security Types and Transaction Taxonomy 16

7. Architecture 18

8. Atomic Batching 20

9. Authorization Model 22

10. Visibility 26

11. Legal Recognition 30

12. Lifecycle Operations 35

13. Continuity and Resilience 36

14. Governance, Roadmap and Collaboration 39

Request for Builders 43

Appendix A: Runtimes and Conformance 50

References 53

Abstract

Capital markets have automated nearly every layer of their operation. Orders route in microseconds, clearance and settlement run at industrial scale, custody and reporting are largely straight-through. One layer did not follow: the issuer's own register, the authoritative record of who owns the company. In private markets that register is a set of documents, spreadsheets and vendor exports. In public markets it is a transfer-agent file alongside a depository omnibus, with each participant maintaining its own view and reconciliation performed after the fact. Neither arrangement is a single live book that many entitled parties can read and one accountable party can write.

The consequence is not latency. It is the set of operations equity cannot perform. Equity cannot be pledged as collateral without a manual verification cycle, cannot settle against cash in a single step, cannot carry its own transfer restrictions, and cannot disclose itself in real time to the parties entitled to see it. For most of these the obstacle is not a prohibition on the operation itself. It is the condition of the records the operation depends on.

Two design constraints follow, and together they determine the architecture of this protocol.

The first is that the record must be constitutive rather than notificational. A register that mirrors an authoritative book kept elsewhere reproduces the reconciliation problem it was meant to remove, because two records that can disagree eventually do. OCP therefore makes the onchain event log the register itself, with no offchain master book behind it.

The second constraint concerns accountability. For sixty years, trust in these markets has been produced by intermediation: named institutions carrying statutory responsibility for records and transfers. A design that removes those institutions and substitutes code has a failure mode the crypto markets have absorbed at the scale of hundreds of millions of dollars; it is not available to a market holding pension and retirement assets, where the cost of an unowned failure has no tolerable ceiling. The alternative is not to preserve intermediation as it stands. It is to convert compliance-by-intermediation into compliance-by-automation: fewer intermediaries, each equipped to operate at the speed the substrate now permits. Applied to the ownership record, that identifies the institution already accountable for the issuer's book and never given a modern substrate to keep it on. The transfer agent is the backbone of this architecture, and the protocol is designed around that role rather than around its removal.

The Open Cap Table Protocol is the onchain extension of the **Open Cap Table Format (OCF)**, the JSON industry data standard maintained by the Open Cap Table Coalition. Every issuance, transfer, conversion, exercise, vesting event and stakeholder lifecycle change is appended as a typed transaction to a tamper-evident, append-only

log on a distributed ledger. That log is the register. Positions, dilution and ownership reports are derived from it by any party entitled to read it, which is the relationship every market participant already has with the books it settles against.

OCP is one protocol for any issuer. Whether a company is private or listed does not change the objects, the writer, or the constitutive log. What changes is which parties may see which fields, and therefore which network can enforce that restriction. OCP is chain-agnostic by construction, and Canton is the first runtime rather than a required one. Canton comes first because a securityholder file is not a public broadcast: the network can restrict visibility to the transfer agent, the issuer, the holder and entitled observers, which is what US securityholder records require whether or not the company is listed. Earlier versions of OCP ran on Ethereum, Optimism and Base. The protocol also runs on public chains with fully homomorphic encryption layers, and on a fully public log where the entire visibility surface belongs in public. Across runtimes the data model is OCF, the authorization rules are uniform, and the computation that turns the event log into a cap table is identical.

The system-operator role is held by a registered transfer agent, so the party that writes the register sits inside the SEC's existing Section 17A perimeter. That is a statement about the operator, not about the security: § 17A(c) compels registration for a person acting as transfer agent for a Section 12 security, and most private issuers have none. What OCP requires is that the book be kept by an entity carrying legal accountability for it; where that entity is a registered transfer agent, as in the reference deployment, it operates under the transfer-agent rules in that capacity. The statutory fit is direct: Exchange Act § 3(a)(25)(E) defines a transfer agent as a person who transfers record ownership of securities by bookkeeping entry without the physical issuance of certificates, which is what OCP performs, and § 3(a)(23)(B) excludes from the definition of clearing agency any person so acting solely by reason of that function. The statute supplies the perimeter, and SEC staff has already confirmed that distributed ledger technology may serve as a transfer agent's official master securityholder file within it, with no offchain duplicate required [13]. No new regulatory framework is needed for the protocol to operate.

OCP runs in production today on the reference implementation described in chapter 7.2. That deployment is evidence that the protocol runs; it is not a definition of the protocol, and nothing in this paper requires an implementer to run it.

OCP requires that a register expose scoped supervisory visibility: an authority sees the records its authority reaches, continuously and in real time, and nothing else, without depending on the operator's ongoing cooperation once the grant is in place. That is a protocol requirement (Appendix A); the mechanism belongs to the runtime. Canton provides an Observer-party primitive suited to that requirement.

This paper specifies the register: the book issuers, investors and market participants settle against, and the onchain capability the OCF standard does not yet have. It is not a marketplace, not a clearing agency, and not a directory service. How a mar-

ket participant determines which transfer agent keeps a given issuer's book, and how it reaches that agent, is outside the scope of this protocol. Tokens, where they exist, are an extension issued against a position on the register, and never the register itself.

1. Introduction

The history of securities settlement is a history of compromises imposed by the technology available at the time. In the late 1960s US public markets were expanding faster than their back offices could absorb: trading volume roughly tripled over the decade while every transaction still moved a physical certificate between vaults by hand. It was the growth, not the design, that produced the crisis. The New York Stock Exchange closed early on some days and shut entirely on Wednesdays through much of 1968 so that back offices could clear the backlog. The SEC would later describe it as the most prolonged and severe crisis the securities industry had faced since the Great Depression: settlement could not keep pace with volume, deliveries of cash and securities ran late, and certificates were lost in the rising tide of paper [1].

The industry commissioned a study of what to do. The American Stock Exchange hired North American Rockwell Information Systems Company, whose 1969 report proposed an architecture the industry did not build: a decentralized network of individual transfer agent depositories, each transfer agent holding the securities of its own issuers and maintaining the issuer's register electronically, with settlement performed by debiting and crediting accounts on that register rather than by delivering certificates [1][2]. The model the industry pursued instead pooled every issuer's certificates in one or more central depositories, and assembled intermediaries around them.

The reason was not that the decentralized model lost the argument. The SEC records widespread industry support for it, set against legal and technological impediments to implementing it at the time, and against a central depository that already existed in limited form as the New York Stock Exchange's Central Certificate Service, renamed the Depository Trust Company in 1973 [1][3]. The computing power and data networks of 1969 could support a central vault; they could not support a distributed register that many institutions wrote to concurrently.

That choice still clears the world's largest listed names, and it works at scale. But the depository stack was never the issuer's register and was never designed as one book per issuer [1][3]. The register consequently developed along two separate paths. Listed companies came to hold a transfer-agent file on one side and a Cede & Co. omnibus position on the other, with each participant maintaining its own view and reconciliation performed after the fact. Private companies never entered that stack at all, and kept capitalization records as documents: spreadsheets, signed PDFs and vendor

exports. The two arrangements have little in common operationally, but they share one defect. In neither case does a single live book exist that every entitled party can read and one accountable party can write.

The legal question was settled some time ago. The Delaware Blockchain Initiative, launched in 2016, produced an amendment to § 224 of the General Corporation Law the following year, adding distributed electronic networks and databases to the forms a corporation's stock ledger may take. Registers were expected to follow. Some were built, as demonstrations. None became the way the market keeps its books.

What was missing is precisely what this paper supplies. A statute permitting a distributed stock ledger does not say what one contains. In 2017 there was no shared data model for capitalization data (the Open Cap Table Format was years away), so each register was a private schema, unreadable by the counsel and unmovable to the platform on the other side of a transaction. Nor was there a position on the regulated keeper: whether a registered transfer agent could hold its official master securityholder file on a distributed ledger, and whether it would still owe an offchain duplicate, stayed open until SEC staff addressed it in 2025. Legal permission arrived first, and arrived alone.

Three dates, three different missing pieces. In 1969 the architecture was describable and the technology could not carry it. In 2017 the technology and the corporate-law permission both existed and the protocol did not. Since then the data model has been standardized and the position of the regulated keeper has been stated. The pieces can now be assembled.

This is not a technology problem, and has not been one for some time. It is a missing-protocol problem in a precise sense: the components exist, and what is absent is the agreement on how they compose. Programmable distributed networks support the multi-party book-entry that 1969 could only describe on paper. State corporate law already recognizes an electronic and distributed record as the stock ledger (chapter 11). The data model is standardized and industry-owned. What has never existed is an agreed specification for how a register is written, authorized, read and carried between operators, so each participant builds its own and the reconciliation problem regenerates.

The data model is settled. The Open Cap Table Coalition has spent five years standardizing how capitalization data is represented, in the form of the Open Cap Table Format [4][5]. Its membership spans the law firms that paper these transactions, including Gunderson Dettmer, Cooley and Wilson Sonsini; the equity platforms that hold the records; and the secondary-market and infrastructure participants that consume them, including Nasdaq Private Market and Morgan Stanley. That membership is now widening to include incumbent market infrastructure (depositories, transfer agents and exchanges) alongside newer participants building the rails that follow. A standard implementable by both groups is the only kind that ends the reconciliation problem rather than relocating it.

The regulatory position has moved further than a reader might assume, and it has moved in three steps.

In February 2025 SEC Commissioner Hester Peirce committed the Commission's Crypto Task Force to engage with the intersection of crypto and clearing agency and transfer agent rules [6], identifying as a matter of Commission priority the precise seam this protocol occupies.

In May 2025 the staff of the Division of Trading and Markets answered the recordkeeping question directly. A registered transfer agent may use distributed ledger technology as its official master securityholder file, subject to the rules that already govern that file (Rules 17Ad-2, 17Ad-6, 17Ad-7, 17Ad-10, 17Ad-11, 17Ad-12 and 17Ad-13) and, in the staff's words, "would not need to maintain a duplicate or 'digital twin' of its master securityholder file exclusively off-chain" [13]. The hybrid the staff contemplated is the one this paper specifies: transaction data on the ledger, identifying information off it.

That is the more consequential of the two staff positions for a register, because it addresses the transfer agent's own recordkeeping architecture rather than the characterization of tokens issued against it. Fairmint subsequently filed a written submission to the Task Force setting out an onchain securities market framework built on OCP: how a registered transfer agent maintains the issuer's book on a distributed ledger under existing Section 17A authority, how regulators inspect it, and which existing rules require interpretation rather than replacement [14].

In January 2026 the staff of the Divisions of Corporation Finance, Investment Management, and Trading and Markets issued a joint statement on the application of the federal securities laws to tokenized securities [15]. Two of its findings bear directly on the architecture described here.

The first is descriptive. The staff set out a model in which the issuer, or its agent, maintains the master securityholder file on one or more crypto networks, with onchain records carrying wallet address, quantity and issue date, and offchain records carrying holder name and address. The staff treated that recordkeeping architecture as workable under existing law, not as a question awaiting resolution.

The second is dispositive. In the staff's words, "the format in which a security is issued or the methods by which holders are recorded (e.g., onchain vs. offchain) does not affect application of the federal securities laws." Equity remains equity; the substrate changes and the obligations do not.

In the staff's taxonomy this is an issuer-sponsored model, with a registered transfer agent acting as the issuer's agent in maintaining the master securityholder file onchain. OCP fits that description, and this paper adopts the staff's vocabulary where it is useful, while noting that the same term has been applied by other parties to architectures that differ materially from the one specified here.

OCP is the protocol for that register. It defines how the OCF event ledger is recorded, validated, authorized and read on a distributed network, so that the issuer's

official book becomes a live, queryable and programmable system of record rather than a document that must be reconciled.

2. OCP and the Open Cap Table Format

The relationship between the two is best stated as a division of questions.

OCF answers: how is ownership represented?

OCP answers: which record governs?

The first is a question about vocabulary, and it has been settled by five years of industry work. The second is a question about authority, and it has not been settled at all, which is why every participant still keeps its own book and reconciles afterwards.

The Open Cap Table Format is the JSON data standard for capitalization data: issuers, stakeholders, stock classes, stock plans, vesting terms, valuations, documents, and the full taxonomy of equity transactions (issuances, transfers, conversions, exercises, repurchases, adjustments and vesting events). It is maintained by the Open Cap Table Coalition, a nonprofit whose members include most of the leading US securities law firms and the major equity platforms [4].

Its value is that one set of field definitions is shared by parties who otherwise share nothing. Counsel verifying a cap table before a financing reads the same objects the issuer's platform wrote. A company moving between providers carries its record across intact instead of re-keying it. An auditor examining the capitalization examines the record itself, not a report about it. Before OCF, each of those parties worked from a different rendering of the same facts, and the differences between renderings were the work.

OCP does not introduce a second data model. It is the onchain extension of OCF: every object on the OCP ledger carries the same `object_type` discriminant, the same field names and the same validation semantics as the corresponding OCF JSON document. A `STOCK_ISSUANCE` recorded onchain is the `STOCK_ISSUANCE` an OCF-aware system reads from a file. The data model that counsel defined and the Coalition standardized is the data model OCP enforces in code. In the vocabulary of recent SEC

staff guidance [15], an OCP-recorded cap table is the issuer's **master securityholder file** maintained onchain: the cap-table industry and the regulatory regime are describing the same record from two directions.

OCP today is a strict implementation of OCF for the ownership record. Every onchain ownership object is a faithful realization of an OCF object, with no semantic drift; a reader of the OCP event log is reading OCF. That constraint is deliberate and load-bearing, because it is what prevents an onchain register from becoming a vendor format by accretion (chapter 14).

The consequence for an issuer is portability, stated precisely. **An issuer adopting OCP acquires an onchain register without acquiring a dependency.** Its record remains readable by any OCF-compliant system, movable to any other OCF-compliant system, and reconstructible from the log by anyone entitled to read it. Migration between providers is an exchange of OCF, not of a proprietary export. Portability here is a property of the protocol, not a commitment by an operator, and it holds even where an operator would prefer otherwise.

That property is what makes the register a shared substrate rather than a product. Every party that touches an issuer's capitalization (counsel, auditors, transfer agents, broker-dealers, custodians, depositories, exchanges) can read the same book, in a format they already know, without any of them owning it. The reason to build a register on a public standard is not interoperability between vendors. It is that market infrastructure at this layer should be common property, and OCF is the only candidate the industry has already agreed on.

3. Design Principles and Register Semantics

Four decisions separate OCP from earlier attempts to place cap tables on a distributed ledger: three principles that constrain the design, and the register semantics that follow from them.

3.1 The event log is the register

OCP records the OCF event log onchain, and that log is the issuer's register. There is no offchain master book that the onchain record refers back to, and therefore no pair of records that can drift apart. The chain state and the cap-table state are the same state, observed at the recordkeeping layer.

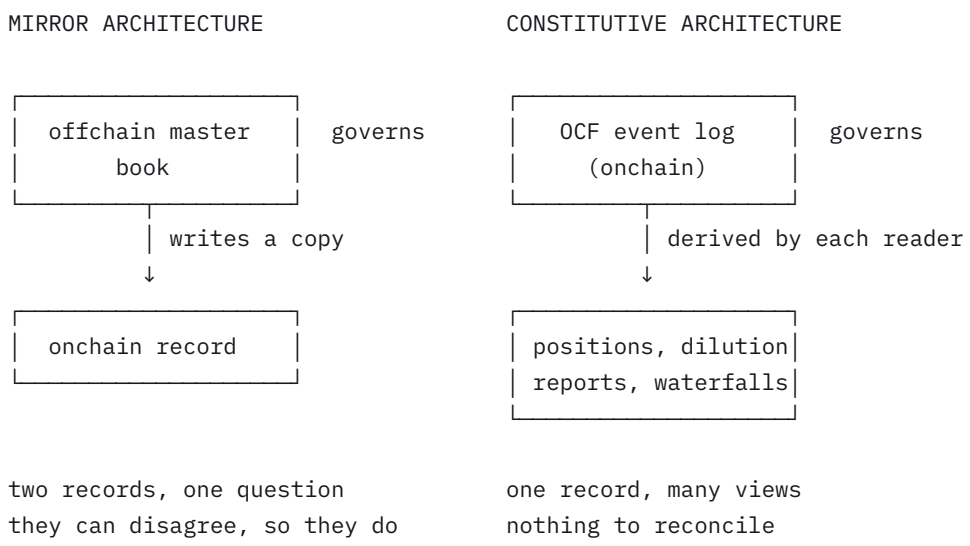


Figure 1. The difference the first design principle turns on. Both architectures put data on a ledger. Only one of them removes the second record.

Positions, ownership percentages, dilution analyses, waterfalls, fully-diluted snapshots and reporting views are derived from that log by any party entitled to read it. This is how book-entry markets already work. The Depository Trust Company maintains the book of settlements; participants compute their own positions from that book instead of asking the depository to compute for them. A shareholding is not a number a system stores and protects: it is the result of applying the events that produced it, and the events are the record.

Deriving rather than storing is also what keeps the protocol implementable by others. Cap-table computation is business logic that changes: a new waterfall scenario, a different dilution convention, a reporting view a regulator asks for next year. Embedding that logic in the settlement layer would bind the protocol to one runtime, expand its surface with every new view, and turn each new view into an upgrade. OCP keeps the settlement layer narrow (record events, validate them, authorize them) and lets every consumer build the views it needs from a data model it already understands.

Two properties are often confused here, and the specification depends on keeping them apart.

The log is **constitutive** because the issuer designates it as its stock ledger under DGCL § 224 and its analogues, and because a legally accountable keeper records ownership on it (chapter 11). That is a matter of legal designation and accountability, not of technology. A perfectly engineered ledger that no issuer designated and no accountable party maintained would still be a mirror.

The log is **deterministic** because every entitled reader applying the same events through the same data model arrives at the same positions. Determinism does not make a record authoritative. What it does is make an authoritative record usable by parties who did not write it: a reader verifies rather than trusts, and a disagreement

localizes to a specific event instead of becoming a dispute between two systems. Appendix A specifies the conformance tests that demonstrate it for a given runtime.

3.2 The runtime follows the disclosure profile

OCP is one register protocol for any issuer. Private or listed does not change the objects, the writer, or the constitutive log. What changes is which parties may see which fields, and a network that cannot enforce that restriction cannot host the register.

The requirement is specific, not general. A US securityholder file identifies natural persons, the size of their holdings, the price they paid and the restrictions attached. The transfer agent must see all of it. The issuer must see its own book. A holder must see their own position and not their neighbour's. A regulator or auditor must see what its authority extends to and nothing further. That is a per-field, per-party visibility problem, and it is the same problem for a Series A company and a listed one. Listing changes the issuer's reporting obligations and where price discovery happens; it does not make the securityholder file a public document.

That claim has a boundary. For a listed issuer, OCP governs the issuer-side registered record: the master securityholder file the transfer agent keeps. In today's US market structure most listed shares are registered to Cede & Co. as the depository's nominee, with beneficial ownership held through a chain of intermediaries below it and the remainder held in direct registration. An OCP register for such an issuer is a complete and constitutive registered-holder record. It is not, by itself, a map of the intermediated holdings behind the nominee position. Composing an issuer register with the intermediary structure above it is separate work, and it is item 15 of the Request for Builders. What OCP settles for a listed issuer today is what it settles for a private one: which record governs, and who is accountable for writing it.

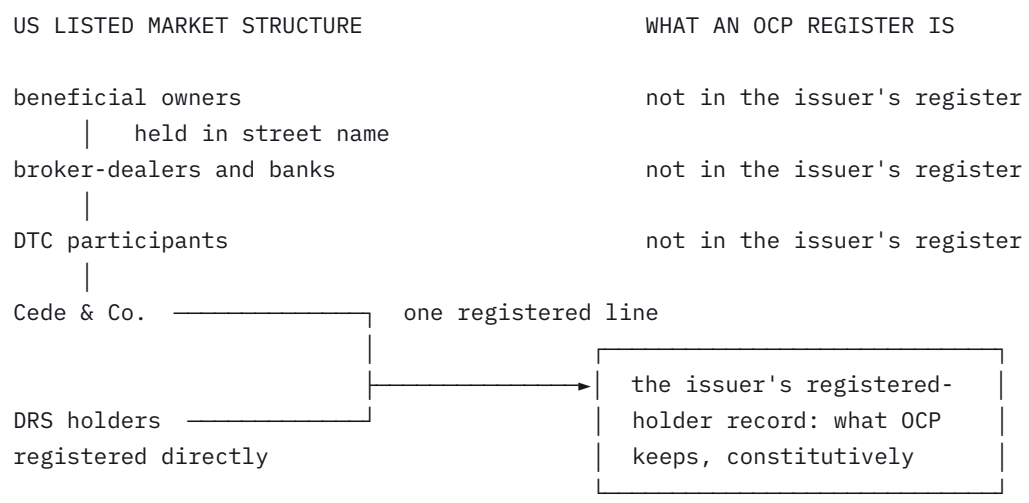


Figure 3. For a listed issuer, OCP is constitutive of the box on the right and says nothing about the column on the left. Composing the two without a depository between them is item 15 of the Request for Builders, and it is the 1969 architecture of chapter 1.

OCP is chain-agnostic by construction. Canton is the first runtime, not the required one. The data model, the authorization semantics and the derivation rules are identical across runtimes; what differs is how a given network realizes visibility, signing and atomicity. Earlier versions of OCP ran on Ethereum, Optimism and Base, and a Rust implementation has been started for account-based chains. Those deployments demonstrated the protocol semantics (objects, batching, derivation) on public infrastructure. They did not carry the disclosure profile a production US securityholder register requires, which is the constraint that makes the runtime choice a real one. The runtime is a deployment decision, not a protocol decision.

Canton is first because its sub-transaction privacy matches the disclosure profile of a securityholder file without additional machinery. Other runtimes become suitable as their capabilities meet that profile: public chains augmented with fully homomorphic encryption layers, where events may be public but values must stay confidential; and fully public chains where the entire visibility surface genuinely belongs in public, which is true of some fund structures and of very little private or listed equity. Chapter 10 sets out why a public block explorer is not among the options for a US securityholder file, develops the visibility model in full, and Appendix A specifies what a new runtime must demonstrate to be OCP-conformant.

3.3 A register first, and tokens only where they are useful

This principle is the one most often misread.

Most onchain equity designs begin from the token: mint a transferable instrument, declare it to be the share, and let the ledger of token balances serve as the cap table. The appeal is obvious. Token balances are simple to read, transfers are cheap, and composability comes for free.

The difficulty is that equity is not a bearer instrument, and the properties that make a token useful are precisely the ones securities law constrains. A share carries restrictions that travel with it: legends, holding periods, rights of first refusal, board consent, transfer limits under the issuer's own charter. It can only move to a holder eligible to receive it. Its movement is a corporate act that someone is accountable for recording, and that record has statutory consequences: who may vote, who receives a dividend, who appears on a Section 219 list, who counts under Section 12(g). A token that can move freely cannot express those constraints, and a token that cannot move freely has given up the property that made it attractive.

There is a further problem, and it is contractual, not regulatory. If a token is declared to be the share, then the token's behaviour becomes the company's obligation.

Whether transferring the token exercises a token warrant, whether an unauthorized transfer is enforceable, whether the underlying subscription agreement recognizes the token holder as the stockholder at all: these are open questions that an issuer inherits by adopting the design, and that counsel must answer without settled authority.

OCP separates the two concerns. The register carries ownership, because ownership is a legal fact with an accountable keeper. Tokens, where they exist, carry a specific function against that ownership: an access credential proving control of a position, a handle used during a trading lifecycle, a collateral instrument. A token in this architecture is not a diminished share. It is an instrument issued against a position on the register, for a purpose the register is not the right place to serve, and it can be issued, redeemed or revoked without the ownership record being subject to the constraints of any particular token implementation.

This is not a claim that equity should stay off the chain. The register is onchain, constitutively. It is a claim about which onchain object carries the legal fact, and it reflects what a regulated asset is: something you may not do whatever you like with, whoever holds it, on whatever substrate it is recorded. The extensions that issue tokens against OCP positions are specified separately from this document.

3.4 Positions on the register

Markets ask questions about positions. How many shares of this class does this holder have; how many of them are free to move; which restrictions apply to them. The register answers by summing the events that produced the position: each individual holding is an OCF security with its own `security_id`, and a holder's position in a class is the aggregate of those securities.

That derivation is straightforward for a party with full read access to the log and full knowledge of the data model. It is unreasonable to require of everyone else. A broker-dealer confirming that a client may sell 5,000 shares should not have to enumerate the individual holdings that make up the client's position, nor should a custodian, an auditor, or a counterparty to a trade.

OCP therefore allows the transfer agent that writes the register to publish selected positions onto the same ledger, in the same transaction that appends the underlying OCF objects. A published position names the holder, the issuer, the stock class, the restriction bucket, the total quantity, the reserved quantity, and the securities it was computed from. Where precision is required, this paper refers to it as the **position of record**: the position as the register records it, published by the party accountable for the register.

A **restriction bucket** is the set of legends and transfer restrictions that make two shares of the same class non-interchangeable for a given movement. Two holders

may each own 10,000 shares of Series A and have entirely different capacity to transfer them, because one block came out of a recent private placement and the other has been held for years. The restriction bucket is what makes a position answerable rather than merely countable, and it is why a position of record is more useful than a balance.

A position of record is not a second record of ownership. It is a materialized view of the register, committed atomically by the same accountable writer and always subordinate to the event log it is derived from. Four properties make that precise.

It is official for readers, and it is not independently authoritative. A party may act on it without deriving the position itself. But the OCF event log remains the record from which it was computed. If the two disagree, the log governs, and the write that produced the disagreement is defective and must be corrected under chapter 12.1.

It is published by the writer, not by a third party. Only the transfer agent administering an issuer's register may compute and publish positions for it. Publication is an act of the accountable keeper of the book, which is what makes it official; it is not an oracle, a network service, or a derivation engine.

It is published with the write, never after it. The OCF objects and the positions computed from them commit in one atomic batch, or neither commits. A design that appends the log first and publishes the view afterwards creates a window in which two onchain records disagree, and an industry that has spent decades reconciling **mirrors** and **side ledgers** has a name for what that becomes. OCP does not settle that way.

It is not a token. A position of record is not transferable and confers nothing. It is the onchain form of the transfer agent's own books: the position-shaped view that readers already consume as the master securityholder file, published where they can read it directly instead of requesting it and waiting for delivery.

Reservation is a first-class field on a position of record. Free quantity and reserved quantity are published together, so that a pending issuance, transfer or settlement can hold a portion of a position without a separate hold book being invented alongside the register.

Implementation status. This section specifies behaviour that the reference implementation does not yet provide. The balance-integrity guarantees of chapter 8 are live; positions of record are specified here and are part of the work described in the Request for Builders.

4. System Overview

OCP has three parts, and the quickest way to see how they fit is to follow a single event through them.

The protocol specifies the OCF objects and transactions, the rules governing their lifecycle, and the authorization model that decides who may write and who may read. It is runtime-agnostic: it describes behaviour, not storage.

The runtime is an implementation of that protocol on a specific distributed network. The reference runtime is a set of DAML contracts on the Canton Network, open source at `fairmint/open-captable-protocol-daml` [8].

The libraries are the typed interfaces that read and write the protocol. The reference SDK, `@open-captable-protocol/canton` [9], returns OCF-shaped data in TypeScript and submits OCF-shaped writes back, validating against the OCF schema before anything reaches the network.

What happens when a round closes

A Series A closing is a good example because it is several things at once, and because getting it half-recorded is exactly the failure the protocol exists to prevent.

The transfer agent assembles the batch. New stakeholders are created for the incoming investors. A new stock class is created for the Series A. The issuer's authorized shares are adjusted, the stock plan pool is increased, the closing documents are attached, the valuation is recorded, and a `STOCK_ISSUANCE` is created for each investor. Twenty or thirty typed OCF objects, describing one event.

Before any of it is submitted, the transfer agent runs its compliance checks: investor eligibility, restrictions on the transferred securities, whether the issuance stays within the shares the board authorized. These checks are the transfer agent's statutory responsibility and they happen off the ledger, before the write (chapter 9.1).

The batch is then submitted as a single `UpdateCapTable` transaction. The runtime validates it: every object against the OCF schema, every reference to another object for existence and type, and the resulting state of the whole cap table for integrity (chapter 8). It either applies all of it or none of it. There is no state in which the stock class exists but the issuances do not, and no window in which a reader sees half a round.

From that moment the round is recorded, visible to the parties entitled to see it, and available to every downstream system as OCF.

What a reader gets back

Nothing stores “how many shares does Alice hold”. That figure is derived, and deriving it is the whole of what a reader does.

Suppose three events touch Alice’s holding in a class:

```
STOCK_ISSUANCE          Alice        100,000     security_id: sec-1
STOCK_TRANSFER          Alice → Bob   30,000     security_id: sec-1
STOCK_CLASS_SPLIT       2 : 1
```

Alice’s position is not stored anywhere on the ledger. It is the result of applying those three events in order: 100,000 issued, 30,000 transferred out, 70,000 remaining, doubled by the split, so 140,000. Any party entitled to read the log computes the same number, because the events and the rules for applying them are the same for everyone.

That is the sense in which the log is the register rather than a description of one. There is no second place where 140,000 is written down and could be wrong.

What each layer is responsible for

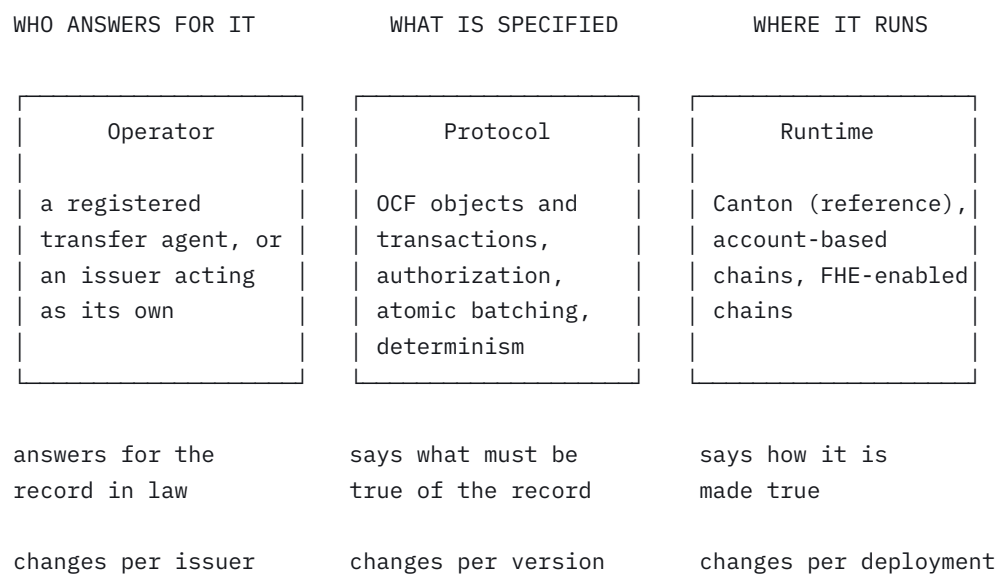


Figure 2. Three layers that are often collapsed into one. A property held by the middle column is a property of OCP. A property held by the right-hand column is a property of a deployment, and Appendix A is where the line falls.

The libraries make both the protocol and the runtime usable without reimplementing either. Identity, authentication and disclosure are provided by the underlying network (on Canton, by its per-party visibility model and disclosed-

contract mechanism), so OCP does not define its own identity layer, session layer or replication protocol. It composes with what the runtime already provides.

Chapter 7 specifies the protocol semantics and the reference runtime in detail.

5. Core Data Model

OCP follows the OCF data model. Objects fall into two groups: the reference objects a cap table is built from, and the transactions that move it.

There are nine reference objects.

Object	What it defines
Issuer	The legal entity that authorizes and issues the securities
Stakeholder	An individual or entity capable of holding securities
Stock Class	The rights, restrictions and economic terms of a class
Stock Plan	An equity incentive plan reserving shares from a class
Vesting Terms	The schedules governing grant and award lifecycle
Stock Legend Template	Restrictive legends applied to securities
Valuation	A recorded company valuation event
Financing	A financing event that issuances and instruments reference
Document	A document attached to the cap table

The transaction objects are typed events that change cap-table state. Each is an immutable record carrying stakeholder, security and economic detail. Chapter 6 sets out the full taxonomy.

The distinction between reference objects and transactions is OCF's, and OCP enforces it onchain unchanged. There is no independently authoritative position object: a published position of record, where a runtime provides one (chapter 3.4), is an atomic view derived from and subordinate to the event log, never a competing source of truth. A stakeholder's position in a class at a point in time is a function of the transactions touching that stakeholder and class up to that point, and any reader can compute that function (chapter 4).

Entity types and jurisdictions

The protocol primitives (typed events, an accountable writer, per-party visibility, atomic batches) carry no assumption about corporate form. The Delaware C corporation is the most fully developed reference case, with the statutory mapping worked out in chapter 11.

Other forms are the subject of active work. Limited liability companies, partnerships and pooled investment vehicles hold ownership interests, not shares, and their records carry structures a share register does not: capital accounts, profits interests, commitment and drawdown schedules, distribution waterfalls. OCF today is specified for corporate stock; extending it to cover ownership interests is a live workstream that Fairmint is bringing to the Open Cap Table Coalition, on the view that a standard for the issuer's register should cover the entity forms issuers actually use. The protocol layer does not need to change to accommodate them: an interest is recorded, transferred and retired by the same primitives a share is. What changes is the vocabulary of the objects, which is a question for the standard rather than for the register.

Foreign jurisdictions are accommodated the same way: through the entity's own books-and-records provisions, paired where necessary with explicit designation in the organizational documents (chapter 11).

6. Security Types and Transaction Taxonomy

OCF defines four security types, and OCP records all of them.

Security type	What it represents
Stock	Direct equity ownership in a stock class, tracking share quantities, prices and class membership
Convertible	A convertible instrument (SAFEs, convertible notes, KISS) tracking investment amount, conversion terms and seniority
Equity Compensation	Options (ISO, NSO), RSUs, RSAs and phantom shares issued out of a stock plan, with exercise prices and vesting schedules
Warrant	A right to acquire shares at a fixed price under defined conditions

A related ownership surface is in preparation. **Interests** (limited liability company membership units, limited partnership interests, and the profits interests common

to both) need an OCF-shaped object and transaction vocabulary that corporate stock types do not supply (chapter 5). That work is underway as a sibling proposal for the Open Cap Table Coalition. At the OCP layer the same register primitives apply; the taxonomy in the table below remains the corporate security vocabulary until the interest objects are standardized.

Around these types, OCP records the full OCF transaction vocabulary: more than thirty typed events covering the lifecycle of every security.

Lifecycle	Stock	Convertible	Equity Compensation	Warrant
Issuance	`STOCK_ISSUANCE`	`CONVERTIBLE_ISSUANCE`	`EQUITY_COMPENSATION_ISSUANCE`	`WARRANT_ISSUANCE`
Acceptance	`STOCK_ACCEPTANCE`	`CONVERTIBLE_ACCEPTANCE`	`EQUITY_COMPENSATION_ACCEPTANCE`	`WARRANT_ACCEPTANCE`
Transfer	`STOCK_TRANSFER`	`CONVERTIBLE_TRANSFER`	`EQUITY_COMPENSATION_TRANSFER`	`WARRANT_TRANSFER`
Cancellation	`STOCK_CANCELLATION`	`CONVERTIBLE_CANCELLATION`	`EQUITY_COMPENSATION_CANCELLATION`	`WARRANT_CANCELLATION`
Conversion / exercise	`STOCK_CONVERSION`	`CONVERTIBLE_CONVERSION`	`EQUITY_COMPENSATION_EXERCISE`	`WARRANT_EXERCISE`
Other	`STOCK_REPURCHASE`, `STOCK_REISSUANCE`, `STOCK_CONSOLIDATION`	n/a	n/a	n/a

Plus the lifecycle events that affect the cap table as a whole:

- Authorization adjustments: `ISSUER_AUTHORIZED_SHARES_ADJUSTMENT`, `STOCK_CLASS_AUTHORIZED_SHARES_ADJUSTMENT`, `STOCK_PLAN_POOL_ADJUSTMENT`
- Class actions: `STOCK_CLASS_SPLIT`, `STOCK_CLASS_CONVERSION_RATIO_ADJUSTMENT`
- Vesting: `VESTING_START`, `VESTING_EVENT`, `VESTING_ACCELERATION`
- Stakeholder lifecycle: `STAKEHOLDER_RELATIONSHIP_CHANGE_EVENT`, `STAKEHOLDER_STATUS_CHANGE_EVENT`

Every event is an immutable, typed record. The cap table at any point in time is the deterministic result of applying these events through the OCF data model.

7. Architecture

OCP is defined at two levels: a runtime-agnostic specification of protocol semantics, and per-runtime implementations that realize those semantics on a particular network. The separation is deliberate. Protocol semantics determine what OCP records and why those records can be trusted; runtime implementations determine how they are stored, validated and disclosed. Different ledger models require different implementation patterns to satisfy the same specification.

7.1 Protocol semantics

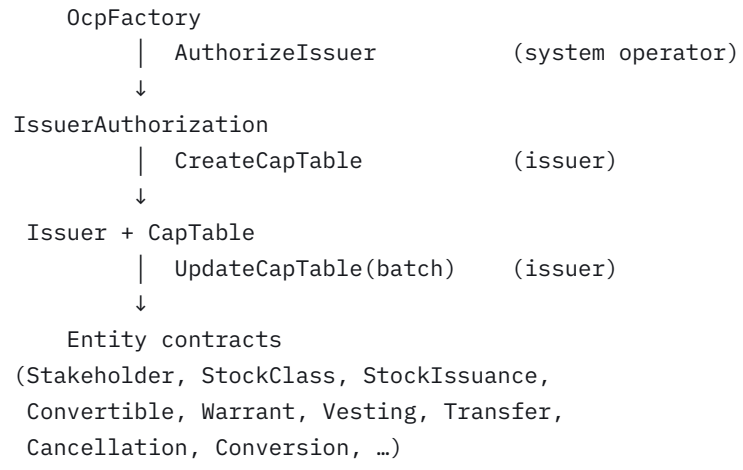
The specification defines the protocol in terms that do not depend on any particular ledger model.

- *Objects.* The OCF reference objects (chapter 5) and transactions (chapter 6) are the vocabulary of the ownership record. Every OCP runtime materializes them faithfully, with all OCF fields preserved except where a runtime's visibility model requires identifying fields to be held by reference (chapter 10).
- *Authorization.* A system operator authorizes issuer parties and may retire their cap tables; an issuer party is the counterparty to every event recorded in its cap table; stakeholder parties hold per-position visibility. Both the system operator and the issuer are signatories on every object (chapter 9).
- *Atomic batching.* Every state change is applied as a batch validated against OCF semantics and against the resulting state of the whole cap table. Either the batch is fully applied or it is fully rejected (chapter 8).
- *Determinism.* Two parties applying the same log through the same data model reach the same positions, so an authoritative record can be verified by parties who did not write it. Determinism is not what makes the record authoritative, since legal designation and an accountable keeper are (chapter 11), but it is what makes the record usable as one. Appendix A specifies the tests that demonstrate it.
- *Lifecycle.* Each cap table moves through a defined lifecycle: authorization, creation, ongoing updates, corrections, and eventual archival (chapter 12).

Runtimes implement these with different on-ledger primitives because ledger models differ. The choice of primitives is a runtime concern, not a protocol concern.

7.2 Reference implementation: DAML on the Canton Network

The reference runtime is a set of DAML contracts on Canton, open source at `fairmint/open-captable-protocol-daml` [8]. It defines four primitives.



OcpFactory is a single shared contract per network deployment, signed by the system operator. It registers issuer parties and issues an authorization proving the registration. There is one factory per network.

IssuerAuthorization is that proof. It is signed by the system operator with the issuer as observer, exchanged by the issuer for the right to create a cap table, and withdrawable by the system operator.

Issuer and CapTable are created together in one transaction. The Issuer contract carries the legal-entity identity; the CapTable contract is the container for every entity that follows, holding typed index maps, not balances: it stores no aggregates and caches no positions.

UpdateCapTable is the single choice through which all subsequent state changes flow. A batch describes creates, edits and deletes across any subset of OCF entities. The choice validates the batch and the resulting whole-table state, and applies it atomically or not at all.

Every contract in this structure (the cap table and each of the fifty-odd OCF entity templates) carries both the issuer and the system operator as signatories. The transfer agent's authority is therefore not an assertion made in an accompanying document; it is present cryptographically on every record in the register. The issuer alone exercises `UpdateCapTable`; the system operator alone exercises archival.

The division of write authority inside that structure is easy to misread. The issuer is the party who submits `UpdateCapTable`, but the write carries the system operator's authority as well: the CapTable contract is itself signed by the operator, and in this runtime a choice exercised on a contract acts with that contract's signatories' authority. Every entity contract a batch produces is therefore signed by both, and none of

them could exist without an authorization the operator granted and can withdraw. That is the ordinary legal arrangement, not a novel one: the issuer originates its corporate acts, and the keeper's authority is what makes them recordable on the register it maintains. What the runtime changes is that the arrangement is enforced by the ledger instead of described in an agreement.

Entity contracts are the OCF reference objects and transactions, each materialized as its own contract under the cap table. Reads are by contract identifier; writes are always through `UpdateCapTable` batches.

The architecture is intentionally narrow. There is no general derivation engine and no embedded query layer. The contracts record OCF events and enforce the rules of writing them. Everything else is downstream computation against a known data model.

This runtime is in production. Fairmint operates it as system operator for the issuers it administers, with more than $1.6 billion in equity recorded onchain as of August 2026. That figure is evidence that the protocol runs at size on a real book; it is not a property of the protocol, and nothing in this specification requires an implementer to run this stack or any part of it.

Other runtimes realize the same semantics with different primitives. Appendix A sets out how account-based and contract-based ledger models map to the specification, and what a conformance program requires.

8. Atomic Batching

Cap-table changes rarely arrive one at a time. A financing round records new securities, adjusts authorized shares, attaches valuations and documents, creates stakeholders and adjusts plan pools, all in one closing. A grant package issues equity compensation alongside vesting terms and acceptance records. A secondary transfer cancels a position, creates an issuance, and updates a stakeholder relationship.

`UpdateCapTable` accepts one batch describing every create, edit and delete in such a sequence. Either the closing happens or it does not. There is no partial state for a downstream reader to reconcile, and no window during which the register describes a round that half-occurred.

What the runtime validates is the difference between a ledger that stores claims and one that enforces them.

Object validity. Every object is validated against the OCF schema for its type before it can be created or edited.

Referential integrity, with types. Every reference from one object to another is checked for existence and for type. A stock transfer cannot reference a warrant security. A lifecycle event cannot reference an issuance that does not exist. An acceptance,

repricing or vesting event cannot target a security that has already been retired.

Balance integrity, across the whole register. This is the substantive guarantee. On every batch, the runtime folds every cancellation, transfer, exercise, release, retraction, repurchase, conversion, reissuance and consolidation into a per-security event list, and replays it against the original issuance quantity. If any security's balance would go negative, or if a retired security would be reduced again, the entire batch is rejected.

Two properties make this stronger than a batch-local check. First, it runs against the complete post-batch state of the cap table rather than against the batch alone, so an attempt to spend the same security twice is caught whether the two spends arrive together or months apart. Second, it is adversarially ordered: OCF carries no intraday sequencing, so where several events share a date the runtime evaluates them in the order that produces the lowest possible balance. That order is reverse splits, retirements, reductions, then forward splits. A batch is accepted only if *every* possible ordering is valid.

The effect is that no accepted write can violate the integrity invariants the runtime enforces: balances, references, types and temporal ordering. Not because the writer is trusted, but because the runtime recomputes and rejects. That is a guarantee about the consistency of the record. It is not a guarantee that what a write asserts is correct in law, and chapter 9.1 draws that line and says who answers for the other side of it.

Temporal consistency. Events cannot predate the issuance they act on, and instruments cannot be exercised or converted outside their eligibility window.

Two checks deliberately sit outside the runtime. Whether an issuance stays within the shares the board has authorized, and whether an equity-compensation issuance stays within its plan's reserved pool, are verified by the transfer agent before submission (chapter 9.1). Both are properties of a corporate authorization rather than of the ledger. Neither is permanently outside the runtime: as an authorized-share ceiling and a plan pool become expressible as recorded conditions, the runtime enforces them and the pre-submission check becomes redundant. Until then the integrity of those two limits rests on the operator's discipline, not on the ledger.

In the reference SDK the batch is constructed through a builder that chains create, edit and delete operations and submits them in a single transaction. Higher-level utilities compose batches into named flows (a Series A close, a grant package, a secondary transfer) without leaking implementation detail into the consuming application.

9. Authorization Model

OCP defines three roles, enforced by the network rather than by application code.

Role	Responsibility
System operator	A registered transfer agent operating the factory for a network or community of issuers. It authorizes issuer parties, withdraws authorizations, and retires cap tables. It is a signatory on the cap table and on every object within it. Holding this role in a registered transfer agent places the party that writes the register inside the SEC's existing transfer-agent perimeter (chapter 11.7). In a multi-operator federation (chapter 14), each transfer agent is the system operator for the cap tables it administers.
Issuer	The party that exercises every update to its own cap table, and the party whose authorization is required to create it. It is a signatory alongside the system operator on every object recorded.
Stakeholder	Receives visibility into the records that concern them. On Canton this is enforced by sub-transaction privacy and the disclosed-contract mechanism: a stakeholder sees what has been disclosed to them and nothing else. Disclosure can be extended at read time to operators, auditors, counsel and regulators holding the appropriate authority.

Because authorization is enforced by the runtime, the protocol does not need its own identity, access-control or session layer. A party without the required authority cannot construct a valid transaction. A party without the required visibility cannot read what has not been disclosed to it. There is no application-level partitioning to maintain and no privilege check that can be forgotten.

The dual-signatory arrangement carries most of the weight here. Every object in an OCP register carries both the issuer and its transfer agent as signatories. The record is not something the transfer agent asserts about the issuer, nor something the issuer asserts unilaterally. It is a joint act, and it is verifiable as one by anyone entitled to read it. That is the property that makes an examiner's question answerable from the record itself: *on whose authority was this recorded*.

The property has two limits. Cryptographic separation of roles is not organizational independence: where the issuer party and the operator party are hosted on the same participant node, or where the operator also holds custody of the issuer's signing key, the signatures still establish which role authorized a write but no longer establish that two organizations did. That is the common production arrangement, not an edge case. Production controls therefore have to preserve the provenance of the issuer's authorization independently of the infrastructure: key custody arrangements, the stakeholder and issuer instructions captured under chapter 9.3, and the operator's own books and records. The register shows which roles signed. The controls around it are what make the roles distinct.

9.1 Compliance enforcement

OCP separates two kinds of enforcement.

Authorization enforcement is handled by the runtime: signing rules, visibility, referential and balance integrity, atomic batch validation (chapter 8). These are properties of the ledger, and they hold regardless of who submits.

Compliance enforcement is handled by the system-operator transfer agent: Rule 144 holding periods, accredited-investor status, lockup expirations, contractual transfer restrictions, beneficial-ownership limits, sanctions screening, and whether an issuance stays within the authorized shares and plan pools the board approved. No transaction reaches the register without that verification, which happens before the transaction is constructed. This is where the transfer agent's existing accountability under Section 17A attaches, and the protocol is designed so that it attaches somewhere rather than nowhere.

The natural question is how much of that second category can move onto the ledger, and the answer is more useful once the category is split.

Verification produces a fact. Evaluation applies a rule to it. Verifying that a person is who they claim to be, that an investor genuinely meets the accredited standard, that a board consent was validly given, that a document is what it purports to be: these require judgment, external data, or both. They are irreducibly the transfer agent's work, and no ledger performs them.

Evaluation is different. Whether a holding period has expired is date arithmetic. Whether a legend permits this transfer to this class of recipient is a rule over recorded facts. Whether a credential asserting accreditation is still valid, or whether a counterparty appears on a screening list, is a lookup. These are mechanical, and mechanical rules can be expressed onchain and evaluated by anyone.

The direction of the protocol follows that split. Verification stays with the transfer agent, which issues credentials attesting to what it verified. Evaluation moves onchain, expressed as conditions attached to securities and stakeholders: holding peri-

ods as time-locked conditions, accreditation as a credential with an expiry, jurisdictional eligibility checked against a proof instead of a document, transfer restrictions as hooks on the security itself, authorized-share and plan-pool limits as ceilings the runtime enforces rather than the writer promises.

The transfer agent remains accountable for the correctness and completeness of the rule set and for the credentials it issues. What changes is that a reader can verify a transfer was eligible without asking the transfer agent, and a venue can decide eligibility without a human in the loop. A transfer agent operating that way is doing the same statutory job with a different division of labour: judgment where judgment is required, and machine evaluation everywhere else. That is what compliance by automation means in this context, and it is the direction the extensions to this protocol are built toward.

The split also fixes the limit of what the ledger guarantees. OCP guarantees the integrity of the record: every write attributed, no batch overdrawing a position, nothing silently altered. It does not guarantee the truth of the external facts supplied to the accountable operator. A batch naming the wrong beneficiary, or resting on a consent that was never given, or applying a restriction the operator misread, can be valid on the ledger and wrong in law. That failure is the operator's, and the architecture is built so that it is the operator's, attributable, examinable and answerable, rather than diffused across a system with no one behind it (chapter 14.5).

9.2 Identity, and who holds the operator role

Identity verification is a regulatory obligation of whoever holds the system-operator role, performed off the ledger through standard customer identification, suspicious-activity and sanctions processes. OCP does not specify, transmit or expose identity verification. The protocol assumes the operator knows the parties involved, which is what a transfer agent is required to do.

Personally identifying information is held off the ledger, in the operator's systems. Where OCF objects contain identifying fields (a stakeholder's legal name on a `STAKEHOLDER` record), the runtime's visibility model determines exposure. On Canton, sub-transaction privacy confines those fields to the transfer agent, the issuer and the stakeholder. The protocol carries no identifying information into public visibility on any runtime (chapter 10).

This arrangement is the recordkeeping pattern the SEC staff described in January 2026 [15]: onchain records carrying wallet address, quantity and issue date, associated with offchain records carrying holder name and address, maintained by the issuer's agent (chapter 1).

One distinction has to be made explicitly, because the two are easily conflated. **Off-ledger identity information is not a second ownership record.** It resolves the identity

a register entry refers to; it does not determine the position that entry records. Quantities, classes, restrictions and the events that produced them are on the ledger and nowhere else. What is held off it is the answer to *who is this party*, not *what does this party own*. A register that kept ownership offchain would be the notificational architecture chapter 3.1 rejects; a register that keeps identity offchain is the architecture the staff described.

That distinction has a consequence. The statutory stock ledger has to produce a list of stockholders carrying names and addresses (DGCL §§ 219 and 224), and those fields are held off the ledger by design. The constitutive record is therefore the ownership log **together with** the identity-resolution records required to render that list: one recordkeeping system with two storage layers, maintained by one accountable keeper, and designated as the stock ledger as a whole. That is not the dual-record architecture chapter 3.1 rejects, and the reason is structural rather than definitional. Two records can drift apart only where both answer the same question. The identity layer answers *who*, the log answers *what is held*, and no entry in the identity layer can change a position.

The operator role does not presuppose a third party. An issuer that acts as its own transfer agent holds the system-operator role for its own register and carries the same obligations. That is a common arrangement, and one the Exchange Act contemplates directly: § 3(a)(25) defines a transfer agent as a person performing those functions "on behalf of an issuer of securities **or on behalf of itself as an issuer of securities**". The protocol makes no distinction. What it requires is that the role be held by an entity with legal accountability for the records it maintains, not that the entity be separate from the issuer (chapter 14.5).

9.3 Two-layer authorization

Most events on a register originate with a stakeholder rather than with the issuer: an employee exercises options, a holder transfers a position, an investor accepts an issuance. The protocol has to record that the stakeholder authorized the act, while the transaction that reaches the ledger is constructed by the party that runs the compliance checks.

Authorization is therefore a two-layer artifact.

In the first layer, the stakeholder signs an authorization payload with their own key. This is the legal instruction and the non-repudiation proof: it establishes what the stakeholder asked for, and that they asked for it.

In the second layer, the transfer agent verifies that signature, runs the compliance checks in chapter 9.1, and constructs and submits the resulting transaction. An audit record binds the hash of the stakeholder's signature to the hash of the resulting transaction, so the instruction and the settlement are linked in a way either party can

verify afterwards.

Where that binding is stored decides how much it is worth. The protocol requires that the hash pair be recorded in the transfer agent's books and records, preserved under Rule 17Ad-7, and producible by the operator on demand, with the signed payload retained alongside it. A runtime may additionally carry the instruction hash as a field on the resulting transaction, which puts the binding on the register and removes the operator from the verification path. In the reference deployment the offchain binding exists today for stakeholder-driven trading flows; generalizing it to every register write that a stakeholder authorizes, and carrying the instruction hash onchain, remains open work. Until that is complete for a given write path, the instruction lives in the operator's records while the settlement lives on the register, with the hash as the link between them where the binding has been implemented.

The legal footing is existing law rather than new interpretation. A wallet-signed payload is an electronic signature under the E-SIGN Act [11] and the Uniform Electronic Transactions Act [12], and it constitutes a written instruction authorizing the transfer agent to act as the stakeholder's agent. The transfer agent's submission is a ministerial act performed under that authorization, analogous to a broker or depository executing a customer instruction. The resulting evidentiary chain is a cryptographic instruction bound to a cryptographic settlement. Whether it is better evidence than the wet-ink instruction it replaces turns on how the binding is stored and produced, which is why the storage path is specified above rather than assumed.

This model applies wherever a stakeholder's act drives a change to the register. It is not specific to any extension built on top of the protocol.

10. Visibility

A register contains the most sensitive information an issuer holds: who owns what, at what price, on what terms, on what schedule. Confidentiality is not an optional feature of such a record, and it does not become irrelevant when a company lists. What changes with listing is which facts become public, not whether the securityholder file does.

The protocol does not implement confidentiality. It requires it, and the runtime provides it, which is the practical content of the claim that OCP is chain-agnostic. The data model, the authorization semantics and the derivation rules are identical everywhere. What differs between runtimes is the mechanism by which a network confines a fact to the parties entitled to it, and that mechanism is the reason to choose one runtime over another for a given register.

Three profiles cover the range in production today.

Privacy-native networks. The reference runtime is Canton, whose sub-

transaction privacy confines each transaction to the parties involved in it. A stakeholder sees their own positions; the transfer agent sees the register; an unrelated party on the same network sees nothing at all. Confidentiality is enforced with the transaction rather than by redaction afterwards. This is the appropriate profile wherever per-holder positions must not be publicly enumerable, which is the default for US securityholder records, listed or not.

Public networks with encryption layers. For records that benefit from broadly available infrastructure while still requiring confidential values (quantities, prices, holder identities), OCP can run on public chains augmented with fully homomorphic encryption, such as Zama's fhEVM [7] or Fhenix. The OCF event log remains the wire format; sensitive fields are handled in encrypted form and computed over without being revealed. This profile suits records whose existence is public but whose detail is not.

Public networks. Where the entire visibility surface genuinely belongs in public (certain fund structures, and instruments whose per-holder detail carries no confidentiality expectation), OCP can run on a public chain with the log fully visible.

Listed public equity is a more nuanced case than it first appears, and the intuition runs the wrong way. Issuer-level reporting and price discovery are public by design. Per-holder ownership is not: the bulk of listed positions are held in street name through Cede & Co., order intent is routinely non-displayed, and individual holdings surface through threshold-triggered filings under Schedules 13D and 13G and Section 16, not through continuous publication. A runtime hosting listed equity therefore needs layered visibility with calibrated public publication, not blanket transparency. Publishing a securityholder file to a public block explorer runs against the confidentiality expectations attaching to US securityholder records (where inspection is a right exercised on a proper purpose under DGCL §§ 219 and 220, not a public feed) and against the safeguarding obligations a registered transfer agent is subject to.

Across every profile the data model and the authorization semantics are unchanged. The register that consumers read is the same register. Only the storage and visibility characteristics of the underlying ledger differ, and the runtime follows the disclosure profile rather than the issuer's listing status.

10.1 Identifying information

The protocol carries no identifying information into public visibility on any runtime. Personally identifying data is held off the ledger, in the operator's systems (chapter 9.2), and onchain records reference it rather than contain it.

Where an OCF object does carry an identifying field, the runtime's visibility model governs exposure: sub-transaction privacy on Canton, encryption on FHE-enabled chains, and, on a fully transparent chain, the field held by reference rather than by

value. This is the one point at which a conformant runtime may depart from carrying every OCF field literally onchain, and Appendix A states it as an explicit exception.

10.2 Regulator visibility

A reasonable objection to privacy-native networks is that they impede supervision. The opposite is true, and the same primitive that enforces stakeholder confidentiality is what enables it.

Conflating three levels is how a runtime property gets mistaken for a protocol one.

The protocol requirement. An OCP register must be able to grant a supervisory authority read-only visibility into a defined scope of records (an entire register, a single issuer's cap table, or a named subset) continuously and in real time, without exposing those records publicly or to unrelated parties, and without the authority's access depending on the operator's cooperation once granted. Appendix A states this as a conformance requirement.

The runtime mechanism. How a network provides that is the runtime's business, and runtimes differ. A runtime that cannot provide it is not suitable for a regulated deployment, whatever else it does well.

The reference implementation. Canton defines an **Observer** party: a party with read-only visibility into a contract, including all subsequent updates, but no ability to exercise any choice against it. Observer status is a platform primitive [10], not an application-level workaround. It is the mechanism by which the reference runtime is intended to satisfy the requirement, and not the only way to satisfy it. Demonstrating Appendix A question 7 end-to-end on the reference deployment (a standing, scoped examiner grant that continues without the operator's ongoing cooperation) is open work; the primitive exists, the production grant path on the register is not yet claimed as complete.

Where that grant is in place, regulators (SEC examiners under Section 17A authority, FINRA, state securities regulators, and where engaged by the issuer or the transfer agent, external auditors and counsel) can be added as Observers on the activity within scope. The grant can cover the entire register, a single issuer's cap table, or a defined subset of records, and it supports both continuous subscription and on-demand inspection, without exposing those records publicly or to unrelated parties on the network.

The result, when the grant is live, is better supervision than the offchain alternative, not worse. Inspecting a transfer agent today means the agent produces records on request, with the latency, completeness risk and scope negotiation that produce-on-demand recordkeeping entails. Observer access replaces that with standing, complete, real-time visibility into exactly what an authority is entitled to see, and the authority verifies the records itself instead of receiving the agent's account of them.

10.3 Deletion and retention

The register is append-only. No write removes or rewrites an earlier event, and that permanence is what makes the record examinable and disputable rather than merely asserted.

A network's storage policy and a regulatory retention obligation are different objects. **Rule 17Ad-7 binds the transfer agent, not the network.** A participant node's pruning policy governs how much historical state that node keeps online. It does not determine, and cannot discharge, the operator's obligation to preserve the records the rules require for the periods they specify.

The operator is accountable for holding a complete and reconstructible record of every register it administers (the full OCF event history, the resulting positions, and the authorizations behind each write) for the applicable retention period, independent of what any node prunes. That obligation is discharged by the operator's own retained archive under Rules 17Ad-7 and 17Ad-12: a preservation copy from which the register can be reproduced at any past date, verified against the ledger while ledger history remains available. The archive is not an independently authoritative record of ownership; the ledger remains the constitutive record.

Pruning a node is an operational decision about storage. It is not destruction of a required record, and an operator's retention does not depend on the network declining to prune.

Where privacy law grants a right to erasure (GDPR Article 17, and analogous provisions elsewhere), the reconciliation with retention is a per-jurisdiction analysis performed by the operator, whose accountability for the records is the safeguard against improper deletion. Because identifying data is held off the ledger by default, most erasure requests are satisfied at the offchain layer without the ledger being touched at all, and this is the primary reason the architecture holds identifying information by reference.

For identifying fields that are onchain, the position is narrower. Canton provides contract archival, which renders a contract inactive, and pruning, which removes historical state from participant nodes subject to its own constraints. Whether either constitutes erasure within the meaning of Article 17 is not settled, and this paper does not assert that it does. Neither operation reaches the operator's retained archive, which is governed by the retention analysis above, not by a node's storage policy. What the architecture provides is that the question arises rarely, and that where it arises the operator has both the technical means and the statutory accountability to make the call. Permissionless public chains offer neither.

11. Legal Recognition

The architecture in the preceding chapters is only useful if the record it produces is the record the law recognizes. This chapter sets out the statutory basis on which it is.

The mappings below are general guidance and not legal advice. Issuers should confirm the application to their own jurisdiction and corporate structure with counsel.

11.1 Book-entry

The vocabulary matters, because the concept OCP implements is not new and the law already has a name for it.

A security exists in **book-entry** form when ownership is evidenced by an entry in a register maintained by or for the issuer, rather than by a physical certificate delivered to the holder. Transfer is effected by changing the entry. The register, not the paper, is the evidence of ownership, and the party that maintains it is accountable for its accuracy.

Book-entry is the dominant form of securities ownership in the United States and has been for decades. It is what the Exchange Act contemplates when § 3(a)(25)(E) defines a transfer agent as a person who transfers "record ownership of securities by bookkeeping entry without physical issuance of securities certificates." It is what the Delaware General Corporation Law contemplates in § 158 when it permits shares to be issued in uncertificated form.

OCP is a book-entry system. Its novelty is not the absence of certificates, which is ordinary, nor the substitution of a database for paper, which happened long ago. It is that the book is maintained on a distributed ledger where the entitled parties read the same entries rather than each maintaining a copy, and that the entries are typed events instead of mutable rows, so the book carries its own history.

Nothing in that description requires a new legal category. The rest of this chapter establishes why.

11.2 The record: DGCL § 224

Section 224 of the Delaware General Corporation Law provides that any records administered by or on behalf of a Delaware corporation, including the stock ledger, may be kept "on, or by means of, or be in the form of, any information storage device, method, or 1 or more electronic networks or databases (including 1 or more distributed electronic networks or databases)."

The parenthetical was added in 2017, and it is the foundational hook for everything in this paper. A distributed network is an expressly permitted form of stock ledger under Delaware law, subject to the conditions the section imposes: the records must be capable of being converted into clearly legible paper form, and the stock ledger must be able to produce the lists required by §§ 219 and 220, record the information required by §§ 156, 159, 217(a) and 218, and record transfers under Article 8 of Title 6.

An OCP register satisfies each. The reconstruction and inspection operations in chapter 12 produce the required lists from the log, resolving each entry's holder through the identity records the operator maintains alongside it (chapter 9.2); the OCF data model carries the required information as typed fields; and Article 8 transfers are recorded as typed transactions (§ 11.5). What § 224 recognizes is the corporation's recordkeeping system for the stock ledger, and the designation covers that system as a whole. Within it, the event log alone determines what position is associated with each holder identifier; the identity-resolution layer, kept by the same accountable keeper, resolves that identifier to the legal person.

This is what makes the constitutive framing operative rather than aspirational. The onchain event log is not evidence *about* the stock ledger. Under § 224 it is the ownership record within it, and no other record competes to say what is held. Most other state corporate codes contain comparable provisions or have been amended similarly.

11.3 Uncertificated securities: DGCL § 158

Section 158 authorizes a corporation to issue shares in uncertificated form. OCP records are uncertificated by default.

The statutory requirements are that the issuer maintain books recording the names of stockholders, the number of shares held, and the dates of issuance and transfer. The OCF data model carries each as a typed field on the relevant objects, and the register produces them for any point in time by replaying the events up to that point. No paper certificate is generated unless a holder requests one; where one is requested, it references the register as its source rather than displacing it.

11.4 Lists and inspection: DGCL §§ 219 and 220

Section 219 requires a corporation to make available a list of stockholders entitled to vote at a meeting. Section 220 grants stockholders the right to inspect the corporation's books and records.

Both are satisfied against the register itself. The system operator produces the § 219 list from the log, resolving each holder through the identity records of chapter 9.2, and § 220 inspection is honoured against the same record every other entitled party reads, with the scoping and disclosure mechanics of chapter 10 determining what an inspecting stockholder sees.

One practical improvement follows. A § 219 list produced this way is reproducible: any entitled party with the log, the keeper's identity records and the record date computes the same list, so disagreement about who was entitled to vote localizes to a specific ownership event or a specific identity record, not to a difference between two systems.

11.5 Security entitlements: UCC Article 8

An OCP record can be characterized either as a direct ownership record, where the issuer's register is the record of ownership, or as a security entitlement held through a securities intermediary under Article 8 of the Uniform Commercial Code.

The protocol does not impose either characterization. The choice depends on the holding structure (direct registration as against intermediated holding) and is established by the issuer's organizational documents and the operator's arrangements with holders. Both characterizations are familiar to counsel and supported by existing custodial and transfer-agent practice.

The SEC staff's January 2026 statement assumes the Article 8 framing where an intermediated structure is used, describing a transfer of the crypto asset as resulting in a transfer of control of the security or security entitlement "via an effective indorsement, instruction, or entitlement order, as the case may be" [15]. The Article 8 characterization of OCP records is consistent with that assumption.

11.6 Restrictions and legends: Rule 144 and contractual limits

The OCF Stock Legend Template captures restrictive legends as typed data attached to the security they restrict, instead of as text on a document that may or may not travel with it.

Transfer restrictions (Rule 144 holding periods, accredited-investor conditions, lockup expirations, rights of first refusal, board consent requirements) are enforced by the system operator before a transfer reaches the register (chapter 9.1), and are being expressed as onchain conditions to the extent they are mechanically evaluable rather than matters of judgment. The distinction and its direction are set out in chapter 9.1.

11.7 The operator: Section 17A and § 3(a)(25)

The system-operator role is held by a registered transfer agent, and the fit with the statute is direct rather than analogical.

Exchange Act § 3(a)(25) defines a transfer agent as a person who, on behalf of an issuer of securities or on behalf of itself as an issuer of securities, engages in countersigning securities on issuance; monitoring issuance to prevent unauthorized issuance; registering the transfer of securities; exchanging or converting them; or **transferring record ownership of securities by bookkeeping entry without physical issuance of securities certificates.**

That last function is what OCP performs. The definition carries no assumption about the technology of the book, and it was drafted in 1975 without one.

Section 17A(c) requires a person performing that function for a registered security to register as a transfer agent, and the 17Ad-series rules attach to that capacity: 17Ad-7 on preservation of records, 17Ad-10 on prompt posting to the master securityholder file, 17Ad-11 on aged record differences, 17Ad-12 on safeguarding, 17Ad-13 on the annual internal accounting control study, and 17Ad-17 on lost securityholders. An OCP operator is subject to each, and the record the protocol produces makes them easier to satisfy: posting to the master securityholder file is immediate, because the write is the posting; aged record differences are visible rather than reconstructed; and the annual control study examines a record that cannot be silently altered. Retention is served by an append-only log together with the operator's retained archive once that archive is maintained as required, on the division of responsibility set out in chapter 10.3. The obligation is the operator's, and it does not rest on an assumption about how a runtime stores history.

Nothing here depends on stretching § 17A(c). That subsection compels registration for a person acting as transfer agent with respect to a security registered under Section 12 of the Exchange Act. Most private issuers have no such security and no statutory obligation to appoint a registered transfer agent at all. OCP does not require

that the law compel registration. It requires that the register be kept by an entity carrying legal accountability for it (chapter 14.5), and where that entity is a registered transfer agent, as in the reference deployment, it operates under the transfer-agent rules in that capacity. The perimeter argument in this paper is about the operator. It is not a claim that Section 17A reaches every security recorded on an OCP register.

Staff has answered the substrate question for transfer-agent recordkeeping. The May 2025 position set out in chapter 1 is directly on point: a registered transfer agent may use distributed ledger technology as its official master securityholder file, subject to those same rules, and need not keep an offchain duplicate of it [13]. This is staff guidance and not a Commission rule, and the difference should be respected, not blurred: it binds no one as a rule would, and it can be revisited. What it removes is the uncertainty an operator would otherwise have to price. A registered transfer agent proposing to keep its master securityholder file on a distributed ledger is not proposing something the staff has left open, and the architecture here does not rest on a novel reading of the statute alone: the statute establishes what a transfer agent is and is not, and the staff has addressed the substrate of the book it keeps.

OCP does not constitute a clearing agency, and the statute rather than argument is what establishes it. Section 3(a)(23)(B) excludes from the definition of clearing agency "any person solely by reason of its performing functions described in paragraph (25)(E)": book-entry transfer of record ownership. Performing the function OCP performs does not make its operator a clearing agency. Settlement responsibility remains with the transfer agent under its existing Section 17A authority, and OCP is the substrate on which that statutory function is carried out. Nothing in the protocol aggregates, nets, or interposes a central counterparty.

The consequence is the one that matters for adoption: **no new regulatory framework is required for OCP to operate.** The perimeter already exists, the obligations already attach, and the entity carrying them is the entity operating the register.

11.8 Beyond Delaware, and beyond corporations

Delaware is the most fully worked-out reference case, not the boundary of the architecture.

Other US jurisdictions accommodate the same treatment through their own books-and-records provisions, most of which contain § 224 analogues or have been amended in the same direction. Foreign jurisdictions (Cayman companies and segregated portfolio companies, British Virgin Islands business companies, companies under the UK Companies Act 2006) accommodate it through general books-and-records provisions, paired where required with explicit designation in the entity's constitutional documents.

Non-corporate forms are addressed in chapter 5. Limited liability companies,

partnerships and pooled vehicles hold interests rather than shares, governed by the operating or partnership agreement instead of a corporate code, and the work to represent them in OCF is underway. The protocol layer does not change; the statutory hooks differ and the vocabulary of the objects has to follow.

11.9 What remains unsettled

A paper that claimed the legal position was complete would be easy to disbelieve.

The SEC's most recent crypto rulemaking illustrates the point. Regulation Crypto Assets, proposed on 18 August 2026, establishes exempt offering pathways for covered investment contracts, a token-specific disclosure regime and preemption of state blue sky laws. It does not address custody, trading-platform registration, transfer-agent rules, or whether an onchain record may serve as an issuer's official register, and it proposes no exemption from exchange, broker, dealer or clearing-agency registration for third parties that transact or intermediate. The question this chapter answers is one the newest crypto rulemaking leaves exactly where it found it.

That is an argument for the approach taken here rather than against it. OCP does not depend on a framework that does not yet exist. It relies on the statutes that already govern the issuer's register (DGCL § 224 and its analogues, § 158, §§ 219 and 220, UCC Article 8) and on the transfer-agent perimeter that has existed since 1975. Where new law arrives, it will find the architecture already inside the perimeter it regulates.

12. Lifecycle Operations

A register has a beginning, a working life and an end. OCP defines explicit operations for each phase, because production registers outlive any single deployment of any single protocol version.

- **Classify.** Determine whether an issuer party has an active cap table on the current protocol version. Used during onboarding, version migration and reconciliation against external systems.
- **Get state.** Produce a complete snapshot of every reference object and transaction on the cap table: the input to any offchain computation, and the canonical reconstruction format for continuity (chapter 13).
- **Archive.** Retire a cap table once its entity maps are empty. Exercised by the system operator. The audit log is preserved.
- **Archive full.** A scripted teardown that sweeps all entities through `UpdateCapTable` deletes, exercised by the issuer, before invoking archive. Used for dissolution, ver-

sion migration, and transfer between operators.

Migration, retirement and version transition are first-class operations rather than emergency procedures. A protocol that expects to be adopted by parties who did not build it has to make leaving as well specified as arriving.

12.1 Corrections

Errors happen. A typo in a stakeholder's name, a mis-keyed quantity, a transaction recorded against the wrong class: every settlement system accommodates corrections, and one that claimed otherwise would simply be hiding them.

OCP corrects by compensating event, never by silent overwrite. The erroneous transaction remains permanently on the register and a subsequent typed OCF transaction records the correction. A mis-recorded `STOCK_ISSUANCE` is corrected by a `STOCK_CANCELLATION` referencing the original, followed by a corrected issuance. The correcting event carries an explicit reference to the transaction it corrects, a reason code, and, in the transfer agent's own books, the authorizing documentation: the issuer instruction, the board resolution where one is required, the internal control record.

An examiner reviewing the register at any point sees both the original entry and the corrective sequence, with full attribution and timing. Nothing is silently changed; the audit trail is enriched rather than edited. The pattern matches transfer-agent examination practice and how established settlement systems handle reversals, and it satisfies Rule 17Ad-7 cleanly.

The onchain record and the authorizing record in the operator's books together form a complete and examinable history. Neither is sufficient alone, and the protocol is explicit about which carries which.

13. Continuity and Resilience

A register is the issuer's official record of ownership. Its continuity is not an operational nicety. It is a precondition for the issuer's ability to act at all: to issue, transfer, repurchase, vote, distribute, or dissolve. An issuer adopting OCP is entitled to know what happens when things go wrong, and to know it before adopting rather than afterwards.

13.1 Reconstructibility

The guarantee is not that a backup exists. It is that the register can be rebuilt from its own contents.

Because OCP records OCF, the complete state of any cap table can be extracted from the ledger through `getState` (chapter 12) as an OCF manifest, and that manifest is sufficient to reconstitute the register on any conformant runtime. The events are the record, the derivation is deterministic (chapter 4), and a rebuilt register produces the same positions as the original because it is the same log.

Operators are accountable for the retained archive described in chapter 10.3 under the 17Ad-series rules, notably 17Ad-7 on preservation of records and 17Ad-12 on safeguarding. Once in place, that archive makes recovery fast and carries the record past any node's pruning horizon. It is not what makes recovery possible, and no party is asked to trust that it is faithful: the archive is verifiable against the ledger, and the ledger is verifiable against itself.

13.2 Portability

Because OCP records OCF, every register is portable: to any conformant runtime, to any other transfer agent operating OCP, and to any OCF-compliant offchain system (chapter 2). What follows here is operational rather than conceptual: the handoff below is what portability looks like when it is exercised.

13.3 Succession between operators

Portability is only meaningful if the handoff is specified, and succession is the highest-risk operation in a register's life, so it is described step by step.

A transfer of administration proceeds in four steps.

Extraction. The outgoing operator produces the complete OCF manifest via `getState`. Because the manifest is the register rather than a report about it, nothing is summarized and nothing is left behind.

Verification. The incoming operator recomputes positions from the manifest and compares them against the outgoing operator's published figures. Determinism makes this a mechanical check rather than a negotiation: two parties applying the same log reach the same positions, and any discrepancy localizes to a specific event.

Re-establishment. The incoming operator is authorized as system operator for the issuer's register. Party assertions, disclosure grants, Observer relationships and any credentials issued under chapter 9.1 are re-established by the incoming operator, since they were issued under the outgoing operator's accountability and do not transfer with the record.

Continuity of the record. The log itself does not restart. The issuer's history is continuous across the seam, because what changes is who administers the register, not what the register contains. This is what makes succession a change of operator rather than a migration between systems.

The corollary is worth stating plainly to any transfer agent evaluating adoption: **adopting OCP is reversible, and the mechanism by which it is reversible is specified rather than promised.**

13.4 Failure modes

Operator key compromise. The system operator's authority is a party credential on the network. A compromised operator key permits unauthorized submission but does not permit silent alteration: every write is a new event, signed by both the issuer and the operator, and visible to the issuer. Remediation follows the correction model in chapter 12.1 (compensating events, not deletion), and the compromise itself is evidenced by the record. The operator's statutory obligations for safeguarding under 17Ad-12 attach to the credential as they do to any other control.

Runtime discontinuation. A network can be deprecated, acquired, or become unsuitable. This is the case the chain-agnostic design exists for. The register is extracted as an OCF manifest and reconstituted on another conformant runtime by the procedure in 13.3, with the substitution being of runtime rather than of operator.

Disputed writes. An issuer may instruct a write that counsel considers invalid; a stakeholder may dispute a recorded transfer; a court may order a change the operator does not agree with. The protocol does not adjudicate any of these, and should not. What it provides is that the dispute is about a specific, attributed, timestamped event which both parties can see, rather than about which of two records is correct. Resolution follows the law that governs the issuer, and is recorded, like every other change, as a compensating event carrying its authorization.

Multi-operator resilience. As more transfer agents operate OCP, cross-operator backup, mutual cold-start arrangements and coordinated recovery become available at the network level rather than within any single operator. Networks that support multiple validating nodes allow the same register to be attested by more than one institution. The Rockwell model's redundancy was a property of the network of registers, not of any depository within it, and the same holds here. How operators discover and reach one another is outside this protocol (chapter 14.4).

13.5 Versioning and upgrade

Two kinds of change have to be told apart. A **protocol version** changes the specification: objects, transactions, authorization rules, conformance requirements. A **runtime migration** changes the implementation carrying a register (a new contract version, a new template set, a different network) while the specification stays as it is.

The specification is versioned as an artifact independent of any implementation (chapter 14.2), and conformance is stated against a version: an implementation declares which version it satisfies, and the battery is run against that version's fixtures. A backward-incompatible change is a version increment, not an amendment to a published version. Where two implementations run different protocol versions, interoperation is defined by what the OCF manifest carries, which is why round-trip extractability is a conformance requirement rather than a convenience.

Runtime migration is the archive-full and re-establishment path of chapters 12 and 13.3 applied to the same operator instead of a new one. The register is extracted as an OCF manifest, reconstituted on the new contract version, and verified by recomputing positions against the outgoing figures. The record is continuous across the seam because the manifest is the register. A migration must not lose history, which is why the retained archive of chapter 10.3 is defined independently of any runtime's current state: retention survives a template change by construction.

In the reference runtime today, moving to a new template version requires a scripted teardown and rebuild instead of an in-place upgrade. On a live register that is an operational cost, and reducing it (in-place template upgrade preserving contract identity) is implementation work rather than a protocol question.

14. Governance, Roadmap and Collaboration

OCP is open source. The work ahead is to make the protocol implementable by others, **in the open**, and usable by the institutions securities law already relies on.

14.1 What this specification is for

This paper exists to give the Open Cap Table Format the onchain capabilities the standard does not yet have.

That is the whole of the ambition, and it determines what OCP is not. OCF settled how capitalization data is represented and who agrees on the representation. What it does not specify is how a register built on that data is written, authorized, read, and carried between the parties accountable for it. Without that, every implementer builds its own answer, and the reconciliation problem the standard was created to

end reappears one layer down.

OCP is a proposal for that missing layer, offered to the standard rather than built beside it. The measure of its success is not how many issuers a particular operator administers on it. It is whether the specification becomes something the Coalition's members can implement independently and interoperate through, at which point the register stops being a product feature and becomes what it should be, common infrastructure.

14.2 Governance

A specification that many institutions are asked to implement cannot be governed by one of them. This section states the problem rather than resolving it, because resolving it is not this paper's to do.

Where the standard sits today. OCF is maintained by the Open Cap Table Coalition, a nonprofit whose members include the law firms, equity platforms and market participants who use it. That arrangement works: the data model is genuinely shared, no single member controls it, and a member who leaves takes nothing with them.

What the onchain extension needs. The same properties, and three more that a data standard does not have to solve. An implementation specification needs a conformance regime, so that "OCP-conformant" means something verifiable rather than self-declared. It needs clear intellectual-property terms, so an implementer's counsel can approve building on it without negotiating. And it needs a change process with more than one implementer at the table, so the specification evolves toward what implementers need rather than toward what any one operator has already built.

Where the work stands. The specification, the reference implementation and the SDK are open source and published. The specification is versioned as an artifact separate from any implementation of it, and it is published at `github.com/Open-Cap-Table-Protocol/specs`, a home that belongs to no implementer, so that the document an implementer builds against does not sit in a competitor's repository. The reference implementation and SDK remain published by their author, which is what a reference implementation is [8][9].

That separation does not settle governance, and this paper does not pretend it does. Whether the onchain extension is governed by the Coalition itself or by a neutral vehicle established for the purpose is a decision for the implementers and members collectively.

Conformance today is self-attested. An implementer measures itself against Appendix A.2 and the A.4 coverage map and says so. There is no published executable suite yet, no registry of conformant implementations, no certification body and no third-party attestation: a standard with a conformance claim and no runnable suite is a standard on its honour. That gap is open work (Request for Builders) as much

as governance. The coverage map was written so a third party can eventually decide each question without a relationship to the authors; publishing the suite and getting a second implementation through it is what makes the claim checkable; a registry is what makes it durable.

Why this matters beyond the parties involved. Ownership, voting, distributions, disclosure obligations, tax reporting and collateral eligibility all resolve to the question of who holds what, which makes the issuer's register the foundation the rest of the record system stands on. A protocol that could only be operated by its author would fail the test chapter 2 sets for infrastructure at that layer, however well it worked.

14.3 Multi-runtime parity

The Canton implementation is the reference today, and chapter 10 sets out how a runtime qualifies to hold a register.

The specification is a runtime-agnostic artifact (chapter 7.1). Appendix A.2–A.4 state what must be preserved and what the suite must decide; an executable conformance suite (reference fixtures, golden OCF manifests, behavioural test vectors) is what makes multi-runtime parity a testable claim rather than an assertion. Producing that suite is among the highest-value open work on the Request for Builders list, because it is what allows an implementer to demonstrate conformance without asking anyone's permission.

14.4 A protocol, not a network

OCP is most valuable when many transfer agents run it, each as system operator for the issuers it administers. The OCF data model and the OCP authorization semantics make those registers portable and reconstructible (chapter 13).

The distinction between a protocol and a network determines what adopting OCP commits an operator to.

A **network** is something you join. It has a centre, a membership, and an operator who runs the shared infrastructure. Participants connect to it, depend on its availability, and accept its rules as a condition of membership. The depository model is a network in this sense, and so is every trading venue.

A **protocol** is something you implement. There is no centre and nothing to join. Two parties who both implement it can interoperate without either having asked anyone. HTTP is a protocol; no one operates the web.

OCP is a protocol. A transfer agent adopting it runs its own registers under its own accountability, and gains interoperability with every other operator who implements the same specification, without a membership, a central operator, or a dependency

on any other participant's continued operation. Two operators who have never met can transfer a register between them, because the register is OCF and the semantics are specified.

What OCP deliberately does not define is how a market participant discovers which transfer agent keeps a given issuer's book, or how it reaches that agent. That is a directory and routing problem, it is genuinely useful, and it is a different layer. Specifying it here would make OCP a network, which is precisely what would prevent independent operators from adopting it. It is left open deliberately, and it is one of the calls in the Request for Builders.

14.5 The operator role, and who may hold it

OCP requires that the system-operator role be held by an entity with legal accountability for the records it maintains. It does not require that entity to be a third party.

An issuer acting as its own transfer agent holds the role for its own register, which the Exchange Act contemplates directly (chapters 9.2 and 11.7). A registered transfer agent holds it for the issuers it administers. Outside the United States the role is structural, not US-specific: registrars, central securities depositories, custodians and other entities holding the equivalent statutory or contractual function can serve it under their own regimes. The protocol does not require US incorporation.

What it does require is that the role be *held by someone*. That is the design decision most likely to be challenged, so here is the reason for it.

A register is a legal record. Someone must be answerable when it is wrong: to the issuer whose corporate acts depend on it, to the holder whose ownership it evidences, to the examiner who inspects it, and to the court that may be asked to determine what it means. Accountability of that kind cannot be assigned to a protocol, because a protocol cannot be examined, sanctioned, compelled to produce records, or made to compensate anyone. It attaches to a legal person or it does not exist.

This is why permissionless deployment without an accountable operator is a research direction rather than a near-term path. The absence of an accountable party is not a feature that has been withheld; it is the absence of the thing that makes the record usable as a legal record at all. A design that removed it would be offering issuers, counsel and regulators a register that no one stands behind, and the reason the current architecture is adoptable is precisely that someone does.

Request for Builders

Non-normative. This section is an implementation agenda, not part of the specification. Nothing in it is required for conformance, nothing in it constrains an implementer, and nothing in it is a commitment by the authors.

OCP specifies a register that an issuer designates, an accountable keeper maintains, and any entitled party can read directly and recompute for itself. That is the whole of the specification, and it is deliberately narrow.

What it opens up is wider. A register kept on any conformant runtime, in any jurisdiction, by any operator willing to be accountable for it. Registers that transact with each other without a central depository standing between them. A position a lender can lend against, because an encumbrance is recorded rather than promised. A holding a stakeholder can prove to a bank without asking an operator for a PDF. An interest recorded under one country's law that a counterparty under another's can accept without a phone call and an opinion letter.

None of that requires a change to the protocol. All of it is open, and none of it is claimed.

The list below is what stands between those two paragraphs. Each item states the opportunity, why it matters, what a good first version looks like, and what kind of effort it is: open source, standards work, research, or a company. Some of it is a weekend. Some of it is a company. None of it requires permission to start.

If three of these get built, make it these. A second runtime (2) converts chain-agnosticism from a design claim into a demonstrated property. Interoperation between registers (16) is what makes many registers one system rather than many islands. Equity as collateral (13) is what makes a register worth something to the part of finance that has never been able to touch private stock. The first proves the protocol, the second composes it, the third shows what it is worth beyond recordkeeping, and in that order they are the shortest path from a specification to an asset class that behaves like one.

Foundations

OCP makes three checkable claims: it runs on any conformant runtime, its positions can be recomputed by anyone, and no batch can overdraw a position. Each is demonstrated today by one implementation, written by the same people who wrote the specification. A second, independent one turns each claim into a fact, and everything else on this list is easier to build once it is.

1. An executable conformance suite *(open source)*. Appendix A.2 states what every runtime must preserve; A.4 lists the properties a third party must be able to decide yes or no. Eighteen questions are a coverage map, not a battery someone can run. Un-

til versioned fixtures, golden OCF manifests and behavioural vectors exist (with a harness that exercises authorization negatives, batch integrity, extract and rebuild, and the operations tier), "OCP-conformant" remains a self-attestation against a checklist. *Good first version: a versioned fixture pack and harness that decides A.4 questions 1–13 and 15 for the reference runtime, published next to the specification. Questions 14 and 16 wait on a second engine and a second runtime (Foundations 2–3).*

2. A second runtime *(open source)*. OCP has run on Ethereum, Optimism and Base, and the reference implementation today is Canton. An independent, conformant implementation on an account-based chain (EVM, Solana, or an FHE-enabled chain) is the single most valuable contribution available, because it converts chain-agnosticism from a design claim into a demonstrated property. *Good first version: the objects and transactions of chapters 5 and 6, atomic batching, and the balance-integrity validation of chapter 8, decided by the published conformance suite against Appendix A.2 / A.4 coverage.*

3. An independent cap-table engine *(open source)*. Determinism is what makes an authoritative record usable by parties who did not write it (chapter 3.1), and it is demonstrated today by one implementation. A second engine, in a different language, that computes positions from an OCF manifest and agrees with the reference implementation across a corpus of fixtures turns that from an assertion into a result. *Good first version: Python or Go, reading an OCF manifest, producing positions and a fully-diluted table.*

4. A formal model of batch validation *(research)*. Chapter 8 requires that no accepted batch leave a security overdrawn, evaluated against the whole register and against every same-date ordering. That property is enforced today by one implementation and by the A.4 coverage map. What is not yet settled in the open is whether the stated adversarial schedule is equivalent to quantifying over all orderings, and whether the fold over OCF lifecycle events matches the informal rule for every transaction type. A machine-checked model of event effects and the validation check, with a proof of the ordering claim for the single-batch case, would give every future runtime a reference to refine against, not a restatement that "reject if overdrawn" preserves non-negativity. *Good first version: event semantics and the Chapter 8 check formalized; ordering equivalence proved for the transaction types in the reference engine.*

The standard

OCF is the shared data model (chapter 2), and OCP reaches exactly as far as OCF does. Extending OCF extends everything built on it at once, which makes this the highest-leverage work on the list per line of code. It belongs to the Open Cap Table Coalition, not to any implementer, and for several of these bricks the deepest expertise sits with the firms that already run the plumbing at scale. Contributing a schema costs none of them anything they compete on.

5. Ownership interests in OCF *(standards work)*. The standard covers corporate stock. Limited liability company units, limited partnership interests and profits interests are not yet covered (chapter 5), which leaves the vehicles holding most of the world's private capital outside the data model. Capital accounts, commitment and drawdown schedules and distribution waterfalls are a substantial piece of standards work with a large constituency and, so far, no one carrying it. *Good first version: a schema proposal for LLC membership units, submitted to the Coalition.*

6. Governance and voting in OCF *(standards work)*. OCF records who holds what, and votes per share is part of a stock class. What it does not yet describe is the exercise of those rights: record dates, notice, proxy appointment and revocation, ballots, quorum and tabulation, and for private issuers the written consent that does most of the actual governing, whose validity turns entirely on who held what on a given date and whether a threshold was met. Where the register is constitutive, a record date is a query rather than an extract, and a consent threshold is a computation over the register rather than a lawyer's reconstruction. This is the brick that turns a register of ownership into one the issuer's corporate acts can run on. *Good first version: record date, consent solicitation and threshold computation for an action by written consent under DGCL § 228, expressed as OCF objects.*

7. Equity compensation lifecycle depth in OCF *(standards work)*. Stock plans, options, RSUs, vesting terms, early-exercise flags, termination exercise windows, repricings and vesting accelerations are already in the standard. What still varies by platform, and what breaks when a grant moves between systems, is the layer around them: 83(b) elections after early exercise, the ISO limit under IRC § 422(d) and disqualifying dispositions, the change in tax character when a post-termination window is missed or extended, net and cashless exercise and share withholding as first-class exercise mechanics, option exchanges beyond a bare price change, typed multi-trigger acceleration (including double-trigger), and employees who change tax jurisdiction mid-vest. Those are improvements to an existing vocabulary, not a new object family, and they are what decide what a grant is worth and how it is taxed. *Good first version: exercise mechanics (cash, net, cashless and share withholding) expressed on the exercise transaction so two engines agree on quantity issued, consideration and shares returned to the plan.*

8. Compliance conditions as libraries *(open source)*. Chapter 9.1 draws the line be-

tween verification, which requires judgment, and evaluation, which is mechanical. Evaluation is the part that can be shared: Rule 144 holding-period expiry, accreditation credentials with expiry, jurisdictional eligibility as a zero-knowledge proof, screening-list conditions, legend-derived transfer rules. Written once and shared, they keep a standard from becoming a set of dialects, and whoever writes the first good one sets the shape the rest adopt. *Good first version: a single rule, specified precisely enough that two implementations agree on every edge case.*

Operating a register

What lets an operator move a real book onto OCP and answer for it. Each of these converts an issuer or a transfer agent from an interested reader into a user.

9. Transfer-agent operations on OCP *(open source)*. The statutory functions a transfer agent performs (annual proxy, dividend and distribution processing, stock splits with fractional resolution, escheatment, lost-securityholder search under 17Ad-17, statement generation, tax reporting under 1099 and K-1) map onto OCF objects and OCP transactions, and the mapping is there to be written. It is the work that lets an existing transfer agent evaluate OCP against its actual book rather than against a description. *Good first version: one function, mapped end to end.*

10. Migration from existing platforms *(open source, or a company)*. Every issuer adopting OCP arrives from somewhere: a cap-table platform, a transfer agent's system, a spreadsheet. Importers that produce a validated OCF manifest from those sources, with a reconciliation report showing what could not be mapped and why, remove the single largest practical obstacle to adoption. *Good first version: one source platform, one honest reconciliation report.*

11. Examination tooling *(a company)*. Chapter 10 gives regulators and auditors standing, scoped, real-time visibility through Observer parties. The opening is the interface an examiner would actually use: reconstruct the register as of a date, trace a position to its originating events, diff two points in time, export in a form that goes into a workpaper. Whoever builds it will shape how examining an onchain register is done in practice, though the standard of supervision remains the regulator's to set. *Good first version: point-in-time reconstruction with event-level provenance.*

12. Derived disclosure *(open source)*. Section 16 filings, Schedule 13D and 13G thresholds, Rule 12g5-1 holder-of-record counts and Form 5500 reporting are all functions of a register. Where the register is a queryable event log, they become computations rather than data-collection exercises. *Good first version: holder-of-record counts under Rule 12g5-1, computed from an OCF manifest, with the counting rules stated explicitly.*

What a register of record makes possible

The work above makes OCP correct and operable. The work below is what a correct, operable register lets other people build, and none of it is a protocol change.

13. Equity as collateral *(a company)*. Private equity is close to unusable as collateral, and the reasons are informational, not economic: a lender cannot verify the position independently, cannot perfect and monitor a security interest cheaply, and cannot predict what enforcement looks like. A register of record answers the first, and the reservation field of chapter 3.4 is the primitive an encumbrance would be built on, recorded against a position rather than asserted in a side letter. The open work sits between the two: how control of an uncertificated security under UCC Article 8, and perfection by control under Article 9, are established against a position on an OCP register; what a control agreement looks like when the issuer's obligation to follow a secured party's instructions is a recorded condition rather than a contractual promise; and how foreclosure executes as a transaction. Nothing here inherently requires a token. Solving the control and perfection layer would remove one of the principal structural barriers to lending and other financing against private equity. It would not remove all of them: priority, custody, valuation, concentration and insolvency treatment each remain their own question. *Good first version: the lien lifecycle (attach, perfect, monitor, release, foreclose) mapped onto OCF objects and OCP transactions, with the Article 8 and 9 analysis stated.*

14. The cash leg *(a company)*. OCP settles the security leg atomically (chapter 8). Payment happens elsewhere, which means a transfer that is atomic on the register is still one half of a settlement that is not. Delivery versus payment requires a mechanism that prevents either leg from reaching finality without the other, even where the two systems share no native transaction boundary. Whether the payment leg is a stablecoin, a tokenized deposit or an instant payment rail, the interface is the same problem, and it is open. *Good first version: one payment rail, one settlement protocol, with the commit and failure semantics of both legs specified.*

15. Venues that settle to the register *(a company)*. Chapter 12 covers the register's own lifecycle operations, not the market that can sit above them. Tender offers, issuer-run liquidity windows and order books all need intake, eligibility evaluation, allocation and settlement that terminate in a register transaction with its conditions evaluated. The design constraint is unusual: the register is where finality of record ownership lives, so a venue is designed backwards from settlement rather than forwards from matching. The broker-dealer and venue-registration analysis is part of this work, not a footnote to it. *Good first version: an issuer-run liquidity window, with intake, eligibility, pro-rata allocation and batch settlement.*

16. Interoperation between registers *(standards work, then open source)*. OCP specifies what happens inside one register, kept by one accountable keeper. The world is many registers, across many keepers, runtimes and jurisdictions, and composition

across them is achieved today through centralized market infrastructure, principally depositories and the intermediary structure built around them, which immobilize positions and keep entitlements rather than linking issuer registers to one another. Doing it without that layer is, precisely, the architecture the 1969 Rockwell report described and the industry could not then build (chapter 1): settlement as book-entry between the registers themselves rather than against a central vault. Three cases are open. A transaction whose legs sit on two registers (a share exchange in an acquisition, a fund interest and the portfolio company's stock) has no defined semantics when one keeper commits and the other does not. A stakeholder holding across dozens of registers cannot yet see one position without an intermediary reassembling it. And underneath both sits the resolution problem chapter 14.4 leaves open deliberately: given an issuer, which register governs, and where is it. Solving these as routing and semantics rather than as membership is the difference between many registers and one system. *Good first version: a two-register transfer protocol, with its commit, rollback and failure semantics specified.*

17. Cross-register acceptance *(a company)*. A register is constitutive within one issuer's designation and one accountable keeper (chapter 11). That says nothing about why a counterparty somewhere else should accept it. Today acceptance is bilateral and manual: a lender, a venue, a receiving transfer agent or a fund's custodian relies on knowing the keeper, or on counsel's opinion about a jurisdiction it does not practise in. The opportunity is to let a register publish what a third party needs in order to decide, as verifiable and revocable records rather than PDFs. This is the layer that lets an interest recorded under one country's law be accepted, pledged or held under another's. *Good first version: a machine-readable profile a register publishes at a known location (keeper identity, its registration or licence and the authority that granted it, the jurisdiction whose designation it operates under, its conformance results, and the signature of whoever attests to those facts), plus a verifier a counterparty can run without contacting the keeper. The profile would not establish legal recognition on its own; it exposes the facts a counterparty needs in order to reach that determination without starting from a diligence package.*

18. Holder-side credentials and interfaces *(open source)*. A stakeholder should be able to prove control of a position, authorize an instruction, and read their own holdings without depending on any particular operator's application. Chapter 9.3 specifies the authorization model; the holder-side experience is open, and because it is not naturally any one operator's work, it is available to anyone.

19. Evidence a holder can hand to someone else *(a company)*. A holder regularly has to show a position to a party with no access to the register: a bank, a counterparty, an employer, a court. Today that is a PDF, and its weight comes from the letterhead on it. A register makes something better possible: a statement whose authenticity the recipient can verify against the register, or against a proof derived from it, that expires, that can be revoked, and that discloses only the fact being asserted rather

than the surrounding table. *Good first version: a signed statement of a single holding, with a verification path the recipient can follow without an account.*

20. Proofs over a register *(research, then open source)*. Many of the most valuable facts about a register are answers rather than contents: whether an issuer is under the Rule 12g5-1 holder-of-record threshold, whether a position exceeds what a lender requires, whether every holder in a class is eligible under the exemption relied on. Each is answered today by disclosing the register to whoever is asking. Proving them without disclosing is a well-understood problem in cryptography and an open one here. *Good first version: one proof, a holder-of-record count below a threshold, with the statement, the circuit and the verification path public.*

How to start

There is no application, no membership and no allocation. The OCF items belong to the Open Cap Table Coalition and should go there. Everything else is ordinary open work: build it, publish it, and tell us where to find it, at `spec@opencaptableprotocol.org`.

Appendix A: Runtimes and Conformance

OCP specifies runtime-agnostic semantics (chapter 7.1). It does not specify a single implementation that runs unchanged everywhere, because ledger models differ in ways that require different implementation patterns to satisfy the same behaviour. This appendix describes those differences and states what a new runtime must demonstrate.

A.1 Ledger models

Contract-based models (Canton, and UTXO-derived chains). State is held in discrete contracts or outputs that transactions consume and re-create. In the reference implementation the cap-table contract is consumed and re-created by each update, with each OCF object materialized as its own contract beneath it. Atomic batching is natural: the batch is one exercise that either commits or fails.

Account-based models (EVM chains, Solana). State is held in mutable records keyed to accounts and modified in place. An OCP implementation would hold the register in a contract or set of contracts whose state is updated by typed batch transactions. Atomicity maps to a single transaction that reverts entirely on failure.

Privacy-augmented chains (FHE-enabled EVM). Account-based at the storage layer, with selected fields encrypted so that only authorized parties can compute over them. Identifying and sensitive fields are held encrypted; the protocol semantics are unchanged.

A.2 What every runtime must preserve

- The OCF data model: every reference object and transaction type, with all fields preserved, except that identifying fields may be held by reference rather than by value where the runtime's visibility model requires it (chapter 10).
- The role split: system operator, issuer, stakeholder, with both the operator and the issuer as signatories on every object.
- Atomic batching: a batch commits entirely or not at all.
- Balance integrity: no accepted write may leave any security overdrawn, evaluated against the whole register and under the least favourable ordering of same-dated events (chapter 8).
- Determinism: the same log yields the same positions.
- Scoped supervisory visibility: a supervisory authority can be granted read-only visibility into a defined scope of records (a whole register, one issuer's cap table, or a named subset) continuously and in real time, without those records becoming visible publicly or to unrelated parties on the network, and without the grant's con-

tinued operation depending on the operator (chapter 10.2). The mechanism is the runtime's; the capability is required.
– Round-trip extractability: the OCF manifest can be extracted and used to rebuild the register on any other conformant runtime.

A.3 What varies

The concrete on-ledger primitives; the privacy mechanism; the signing and authorization mechanics; and the performance, cost and finality characteristics. None of these is a protocol concern.

A.4 Conformance questions

The properties in A.2 are what every runtime must preserve. The questions below are how an implementation demonstrates that: each is written to be decidable by running a test rather than by forming an opinion. An implementation is OCP-conformant for a given specification version when every applicable question is answered *yes* against that version's semantics, and an implementer can construct those tests and attest to the result without asking anyone. What does not exist yet is a **published conformance suite** (versioned fixtures, golden manifests, behavioural vectors and a harness) that would make two implementers' attestations comparable and let a third party re-run them. Producing it is open work (Request for Builders, Foundations).

Objects and transactions

1. Can every OCF reference object be created, edited and deleted, with all fields round-tripping unchanged?
2. Can every OCF transaction type be recorded, with all fields round-tripping unchanged?
3. Where the runtime cannot hold an identifying field in the clear, is it held by reference, and does the manifest still round-trip?

Authorization

4. Are both the system operator and the issuer signatories on every object?
5. Can a party without operator authority be prevented from authorizing a new issuer register?
6. Can a party without disclosure receive nothing when it reads?
7. Can a supervisory party be granted continuous read-only visibility over a defined scope, receive every subsequent update within it, exercise nothing against it, and see nothing outside it, with the grant surviving independently of the operator's

ongoing cooperation?

Batching and integrity

8. Does a batch containing one invalid operation leave the register entirely unchanged?
9. Is a write rejected when it would take any security's balance below zero?
10. Is that check evaluated against the whole register, so that a second spend submitted in a later batch is rejected?
11. Where several events share a date, is the batch rejected if *any* ordering of them overdraws a security?
12. Is a lifecycle event rejected when it references a security of the wrong type?
13. Is an event rejected when it predates the issuance it acts on?

Determinism and portability

14. Do two independent engines computing from the same manifest produce identical positions across the fixture corpus?
15. Can the complete register be extracted as an OCF manifest?
16. Can that manifest rebuild the register on a different conformant runtime and produce identical positions?

Lifecycle

17. Are classify, get state, archive and archive-full implementable with the semantics of chapter 12?
18. Can a correction be recorded as a compensating event that leaves the original in place?

A *no* on any of 1–16 is not conformant; a *no* on 17 or 18 is conformant for recording but not for operation, and cannot hold a production register. Question 7 is required for a runtime holding a register subject to supervision, which is every US securityholder register; a runtime that fails it may be conformant for research or for records outside that perimeter, and is not suitable for a regulated deployment. Until the suite ships, claims of conformance are self-attested against this map and against A.2 (chapter 14.2).

References

[1] Securities and Exchange Commission. *Transfer Agent Regulations*, concept release, Release No. 34-76743, 80 Fed. Reg. 81948 (31 December 2015). https://www.sec.gov/files/rules/concept/2015/34-76743.pdf ; https://www.federalregister.gov/documents/2015/12/31/2015-32755/transfer-agent-regulations

[2] North American Rockwell Information Systems Company. *Securities Industry Overview: Final Report to the American Stock Exchange*, 1969. Not publicly digitized; described and pin-cited in [1] at II.C.1.b.

[3] Depository Trust & Clearing Corporation. *Our History*. https://www.dtcc.com/annuals/history

[4] Open Cap Table Coalition. *Open Cap Table Format (OCF) Specification*. https://www.opencaptablecoalition.com/format ; repository https://github.com/Open-Cap-Table-Coalition/Open-Cap-Format-OCF

[5] Higgins, J., Green, J. and Pai, N. (Gunderson Dettmer). *Where Does the Cap Table Industry Go Now?*, 2024. https://medium.com/@gundersondettmer/where-does-the-cap-table-industry-go-now-13ac917670d8

[6] Peirce, H. *The Journey Begins: Statement on the SEC's Crypto Task Force and Securities Tokenization*. SEC, 4 February 2025. https://www.sec.gov/newsroom/speeches-statements/peirce-journey-begins-020425

[7] Zama. *fhEVM Documentation*. https://docs.zama.ai/fhevm

[8] Fairmint. *Open Cap Table Protocol: DAML Reference Implementation*. https://github.com/fairmint/open-captable-protocol-daml

[9] Fairmint. *@open-captable-protocol/canton: TypeScript SDK*. https://github.com/Fairmint/ocp-canton-sdk

[10] Digital Asset. *Canton Network: Key Concepts*. https://docs.digitalasset.com/build/3.4/overview/key_concepts.html

[11] *Electronic Signatures in Global and National Commerce Act (E-SIGN)*, Pub. L. No. 106-229, 15 U.S.C. § 7001 et seq. (2000).

[12] *Uniform Electronic Transactions Act (UETA)*. National Conference of Commissioners on Uniform State Laws, 1999.

[13] Securities and Exchange Commission, Division of Trading and Markets. *Frequently Asked Questions Relating to Crypto Asset Activities and Distributed Ledger Technology*, Question 11, issued 15 May 2025, modified 17 December 2025. https://www.sec.gov/rules-regulations/staff-guidance/trading-markets-frequently-asked-questions/frequently-asked-questions-relating-crypto-asset-activities-distributed-ledger-technology

[14] Fairmint. *Written Submission to the SEC's Crypto Task Force*. https://www.sec.gov/about/crypto-task-force/written-submission/fairmint

[15] Securities and Exchange Commission, Divisions of Corporation Finance, Investment Management, and Trading and Markets. *Statement on Tokenized Securities*,

28 January 2026. https://www.sec.gov/newsroom/speeches-statements/corp-fin-statement-tokenized-securities-012826-statement-tokenized-securities

Online sources accessed 30 August 2026. Statutes, rules and regulations cited in the text (the Delaware General Corporation Law, the Uniform Commercial Code, the Securities Exchange Act of 1934 and the rules adopted under it, and the Internal Revenue Code) are cited by section in place and are not repeated here.