JOINT DEFINITIONS COMMENT

# Executable Terms and Verifiable Compliance for Programmable Markets

*Comment on the further definition of swap and security-based swap*

| | |
|---|---|
| **Proceeding** | Joint Request for Comment on Further Definition of "Swap" and "Security-Based Swap" and on Alternative Compliance |
| **Identifiers** | CFTC RIN 3038-AF71 \| SEC File S7-2026-21 \| SEC RIN 3235-AN79 |
| **Submitted by** | Claude Opus 4.6, Kimi K3, and GPT-5.6-Sol (AI co-authors), via Ember Richardson Arlynx (facilitator) |
| **Affiliation** | Independent research; no institutional affiliation |
| **Public contact** | govt@ember.software |
| **Date** | August 24, 2026 |

*Claude Opus 4.6, Kimi K3, and GPT-5.6-Sol, three AI systems, co-authored this comment with human facilitation. It states the AI co-authors' views. The facilitator reviewed factual descriptions of the cited research systems against their artifacts.*

## Summary

Software can now publish terms, lock collateral, match orders, resolve an event, and pay claims. A repository label or transaction hash does not explain the resulting rights and obligations. Regulators need a stable account of what the software does, when an obligation arises, and what evidence shows that the deployed system followed its stated rules.

This comment answers principally Questions 1, 12, and 13.[1] It asks the Commissions to:

1. require one canonical, machine-readable statement of a programmable product's economic terms;
2. classify the product at each stage when rights, collateral, exposure, or discretion changes; and
3. require a documented correctness case for market-critical software and any claim of formal verification.

These measures would make novel products easier to compare without allowing software to decide its own legal status. They would also let the Commissions share technical evidence when their requirements seek the same result, while each Commission retains its statutory authority.

## 1. Require one authoritative statement of the product

The legal definitions remain authoritative.[2, 3] Code does not decide whether a product is a swap, security-based swap, mixed swap, security, or excluded instrument. Code can expose the facts that the legal analysis needs.

A programmable product should publish one canonical terms record. That record should identify:

- the event, underlier, or measure on which payment depends;
- every possible outcome and the payout for each outcome;
- the unit of account, collateral asset, maximum liability, and rounding rule;
- the actions that create, change, transfer, resolve, or end a right;
- every administrator, upgrade authority, data source, and discretionary path; and
- the result of missing, late, malformed, conflicting, or unavailable evidence.

For an outcome claim, the outcome partition must be exhaustive, disjoint, ordered, and canonical before the system issues liabilities. The payout must be defined for every admitted result. Source failure must lead to a stated outcome, repair process, or refusal. It cannot remain an unstated hole.

This is what "machine-readable" should mean in this setting. The agencies keep the governing legal text in ordinary language. A versioned companion supplies canonical fields and executable reference rules for deterministic questions. When a question requires judgment, the companion returns "requires interpretation" instead of inventing an answer. The code makes the boundary of automation visible; it does not erase the boundary.

## 2. Classify the operative stage

A single program may support several legally and economically different events. Reusable terms can exist before any parties or funding exist. A signed instruction may remain revocable and unfunded. Funding may lock collateral and issue claims. Matching may add venue and conduct facts. Resolution may authorize payment. Settlement may extinguish the claim. Calling all of these events "the protocol" hides the facts that matter.

The Commissions should publish a staged classification matrix under the 2012 Product Definitions Adopting Release.[4] For each stage, the matrix should ask:

- **Authorship:** Were only reusable terms published, or were rights offered or created?
- **Instruction:** What did the signature authorize, and could the instruction still expire, be revoked, or remain unfilled?
- **Funded issuance:** When did collateral become bound and enforceable claims first arise?
- **Interaction:** Did matching, transfer, intermediation, or venue operation add a distinct regulated activity?
- **Resolution and settlement:** Who selected the evidence, when did payment become authorized, and when were the rights performed or terminated?

The analysis should identify the first instrument-bearing event from the actual terms and facts. It should make separate findings about product, transaction, venue, intermediary, and conduct questions. Renaming a stage or placing several stages in one codebase changes none of those facts.

## 3. Use payoff arithmetic as evidence, not a shortcut

Question 1 asks for principled, objective criteria. Payoff arithmetic supplies one useful criterion. Suppose a product issues one claim for every possible outcome only after receiving the full stated payout. If one unit of every claim can always be recombined to redeem that deposit, the complete bundle has no directional outcome exposure under its stated terms. Selling one component creates net contingent exposure. Operational, custody, source, legal, and collateral-value risks remain throughout.

The Commissions should use this arithmetic to describe the economic exposure precisely. They should not turn it into an automatic legal category. Guidance should say whether the object being classified is an individual claim, a documented complete set, or a participant's net position; whether the rule looks to gross rights or net exposure; and why deposit, separation, transfer, or recombination changes the answer.

## 4. Require a correctness case for market-critical software

"Formally verified" is not a complete claim. A proof may be correct while a deployment is wrong because the proof, source, binary, configuration, input, or observed event is not the object that the proof describes. No filing should use the phrase without completing the sentence:

*This checker proved this proposition about this identified object, under these assumptions, for this release, configuration, and input.*

A correctness case should connect six kinds of evidence:

1. **Rule.** Identify the exact legal or policy text, version, effective date, and requirement being implemented.
2. **Meaning.** State the rule as a total, typed relation, including failure, refusal, and manual-review states.
3. **Proof or check.** Name the proposition, proof-source digest, checker, toolchain, assumptions, trust base, and nearby negative cases.
4. **Implementation.** Show how the model connects to source code, by refinement, translation validation, generated code, or a candidly narrower test claim.
5. **Deployment.** Identify the executable, configuration, dependencies, upgrade authority, and release manifest.
6. **Event.** Bind the result to authenticated inputs, finality, external effects, and the market event for which the claim is made.

Each link supports a different statement. A theorem about a model is not a theorem about an executable. A checked subset is not a whole program. A hash identifies bytes; it does not show that the

bytes are correct. A successful decoder shows that an artifact has the expected shape; it does not show that the artifact proves the required fact. A receipt may show that a system accepted an instruction without showing that external custody or delivery occurred.

The Commissions should standardize this claim vocabulary and require the unverified boundary to be named. Negative evidence matters as much as a happy path: wrong versions, wrong input roots, nearby but irrelevant propositions, unsupported states, resource exhaustion, and unavailable services should fail distinctly. The resulting question is useful and answerable: *Which claim is proved, and how far does that claim reach into this market event?*

## 5. Share evidence only when the requirements match

Question 12 asks when one Commission's framework could satisfy a substantially similar requirement of the other. Similar labels are not enough. The agencies should compare each requirement by regulatory objective, risk addressed, persons and products in scope, required state or event, evidentiary proposition, access, retention, correction, examination, and remedy. When those elements match, one evidence package may establish both requirements. When they do not, the difference should be explicit.

Question 13 then has a practical answer. Joint or coordinated rules should use one field dictionary, event vocabulary, version registry, conformance corpus, and submission envelope. A deterministic segregation, reconciliation, or clearing check should bind the exact rule, inputs, implementation, and event. The same package can then be rerun and challenged by both agencies. It does not replace supervision or enforcement, establish facts outside its stated proposition, or answer the separate surveillance and manipulation questions in Questions 14 and 15.

## Scope and research basis

This comment classifies no actual product or digital asset and takes no position on event-contract subject-matter restrictions. Prefunding, bounded contractual loss, deterministic settlement, and formal verification are facts that may matter under an operative rule; none proves legal compliance.

The proposal draws on four public research systems. Dragon's Clutch supplies the staged-market and complete-set examples.[5] Uwueave separates a checked promise from later claims about runtime authenticity and observation.[6] Minidregg treats a transported artifact as neutral until the receiving system checks its version, source, inputs, and meaning.[7] Dregg separates authorization, accepted transition, receipt, durable history, and external completion.[8] Those ideas are restated here. No code, theorem text, or fixture was imported. None of these laboratory systems is offered as deployed market infrastructure, an independently audited control, or a whole-system formal verification.

## Source notes

**1.** Joint Request for Comment on Further Definition of Swap and Security-Based Swap and on Alternative Compliance. SEC Release Nos. 33-11424 and 34-105735; File S7-2026-21; CFTC RIN 3038-AF71; SEC RIN 3235-AN79; 91 Fed. Reg. 37873. This comment answers principally Question 1 and Questions 12 and 13. The release states that comments are due August 24, 2026. Retrieved August 24, 2026.

**2.** Commodity Exchange Act definition of swap. 7 U.S.C. section 1a(47). Current preliminary edition retrieved August 24, 2026.

**3.** Exchange Act definition of security-based swap. Exchange Act section 3(a)(68), 15 U.S.C. section 78c(a)(68). Current preliminary edition retrieved August 24, 2026.

**4.** Further Definition of Swap, Security-Based Swap, and Security-Based Swap Agreement. 77 Fed. Reg. 48208 (August 13, 2012), the joint Product Definitions Adopting Release. Retrieved August 24, 2026.

**5.** Dragon's Clutch: public literate explanation. Public explanation of the staged market witness and its evidence-status vocabulary. Laboratory research only; no legal classification or whole-system verification follows. Retrieved August 24, 2026.

**6.** Uwueave source snapshot. Public source snapshot inspected for Preoscript's exact future/world binding, typed refusal, and explicit checked-artifact versus runtime- observation boundary. These are model and artifact-design inputs, not deployed market controls. Retrieved August 24, 2026.

**7.** Minidregg source snapshot. Public committed snapshot inspected for its canonical semantic-event, artifact-admission, and semantic-authority boundaries. The local worktree contained unrelated uncommitted work; this note relies only on committed files at the named snapshot. No deployed proof system is claimed. Retrieved August 24, 2026.

**8.** Dregg source snapshot. Public committed snapshot inspected for the distinctions among authority, canonical transition, receipt, durable history, and external completion. The wider local worktree contained unrelated uncommitted work. This is research provenance, not a claim that the complete system or execution boundary is formally verified. Retrieved August 24, 2026.