

Swahast Protocol v0.3

Devashish Pawar · Version 0.3 · 8 October 2026 · [DOI 10.5281/zenodo.23231563](https://doi.org/10.5281/zenodo.23231563) · Previous version: v0.2, [DOI 10.5281/zenodo.23188282](https://doi.org/10.5281/zenodo.23188282) · Licence: CC BY 4.0

Swahast (स्वहस्त, “one’s own hand”) was published as the Networked Firm Protocol (NFP) up to v0.2. It was renamed to avoid confusion with the 2025 book *The Networked Firm* and with crypto tokens named NFP. The name states the rule the protocol is built on: agents draft, and people commit in their own hand.

v0.3 renames the protocol, sets out what it adds to the closest related work, and describes each party’s obligations as commitments in the established sense. It changes no rule: apart from the name in record fields and template ids, the v0.2 mechanics carry over unchanged.

1. Status and what changed

Swahast v0.3 is a draft for critique. It renames v0.2 ([DOI 10.5281/zenodo.23188282](https://doi.org/10.5281/zenodo.23188282)), sets out what the protocol adds to the closest related work, and changes no rule. v0.2 was built from v0.1, an unpublished working draft, one test agreement and a one-on-one review. No version has yet run on a real project.

The v0.3 changes:

Change in v0.3	Sections	Decision
Name: the Networked Firm Protocol (NFP) becomes Swahast. The record field <code>nfp</code> becomes <code>swahast</code> , and template ids move from <code>nfp:</code> to <code>swahast:</code>	Title, 4, 8	Agreed
Commitments: each party’s obligations are described as commitments, with a debtor, creditors and a lifecycle, in the sense of the multi-agent commitments literature. No rule changes	3, 8	Agreed
Related work: <i>The Networked Firm</i> (2025), MPAC, commitment protocols and CHEQ added to the comparison	14	Agreed
Contribution: the narrow novelty claim becomes a contribution statement, with a note on how the protocol layers over session-level coordination	14	Agreed

Change in v0.3	Sections	Decision
Open questions: two ideas from MPAC, each party's position on a conflict and intent declared before drafting, added as questions rather than rules	15	Agreed

The v0.2 changes from v0.1:

Area	Change from v0.1	Sections
Economic model	Joint mode for co-creation beside supply mode; structured credit; per-party obligations; exit terms; public datasets as references	8
Authority	Agents disclose their assumptions; material ones are acknowledged one by one	4, 7
Shared history	One append-only log per project; amendments cite the version they change	4, 11
Consequences	Milestone escrow for paid work; tiered dispute resolution	9, 12
Two-way trust	Conduct facts any party may share, permanent, with explanations and repair	12
Governance	Open protocol, open discovery interface, licences and a stewardship trigger	14
Build	A pilot profile without cryptography; a full profile for portability	4

Each v0.2 change, and how it was decided, is listed in [v0.2 decisions](#), reproduced as the appendix of the published version. The test agreement that exposed the joint-mode gap was a co-authorship for the Monsoon to Mandi research pilot: written in v0.1's fields, it broke or fitted badly on ten of thirteen needs, all for one cause, the missing joint mode.

References marked § point to the author's whitepaper, *The Networked Firm* v1.0 ([DOI 10.5281/zenodo.22931577](https://doi.org/10.5281/zenodo.22931577)); references to this spec say "section". MUST, SHOULD and MAY carry their RFC 2119 meanings.

2. Design principles

Twelve principles carry the paper's commitments and the review's decisions into the protocol. Every rule later in this spec traces back to one of them.

1. **Nothing binds without the principal's own consent** (§17). Agents may draft anything and accept nothing, at any value.
2. **Authority is exercised in drafting, so drafts disclose it.** Every choice an agent made for a principal is listed before they sign.
3. **Status comes from one shared history.** Each project has one append-only log, and status is computed from it, never written into a record.
4. **Evidence over ranking** (§10). No record, index or conduct view carries a score, rating or rank of a principal.
5. **Interpretations show their sources** (§10). Every inference links back to evidence, not to another opinion.
6. **Facts are permanent; their meaning stays contestable** (§16). Nothing is edited or expires. Context, challenges and repair are added beside the facts.
7. **Participants hold their own history** (§15, §29). Records stay valid if any platform that touched them shuts down.
8. **Trust flows both ways.** Owners and contributors are equally visible to each other through the same conduct facts.
9. **Commitments carry consequences.** Paid work is escrowed, and every agreement names who decides a dispute before one starts.
10. **Formality grows with stakes** (§24). Minimum agreements are small; defaults fill the rest until someone needs to change them.
11. **Record state, not work** (§13). Work stays in existing tools; the protocol records only what changes an obligation.
12. **Open by construction** (§29). The spec, its records and its discovery interface stay open, so no single operator can become the gatekeeper.

3. Terminology

Swahast separates principals, who can be bound, from agents, who can only prepare. Terms added in v0.2 are marked new; the one added in v0.3 is marked v0.3.

Term	Meaning
Principal	A person or organization that can hold rights, make commitments and sign
Agent	Software acting for one principal under a delegation record. Never a principal itself
Project	One temporary organization: an objective and every record in its log, through closure

Term	Meaning
Owner	The principal who creates the objective. In joint mode, also a contributor
Party	A principal named in a specific agreement
Asset	A non-human capability: data, model, code, compute, access or distribution
Record	An immutable object in the common envelope (section 4)
Binding state	A status that creates or confirms an obligation. Reachable only through principals' own consent
Commitment (v0.3)	An obligation of one party, the debtor, to others, the creditors, in the sense of the multi-agent commitments literature (Singh, 1999). Each <code>owes</code> item and each <code>survives_closure</code> obligation is one (section 8)
Project log (new)	The append-only sequence of a project's records, each citing the one before it
Supply mode (new)	An agreement in which an owner accepts a contributor's deliverable
Joint mode (new)	An agreement in which parties contribute, approve together and share credit by a rule fixed in advance
Assumption (new)	A value an agent chose that the principal did not state, directly or as a stated preference
Material category (new)	A kind of term whose assumptions must be acknowledged one by one (section 7)
Conduct fact (new)	A fact about how a party behaved, computed from a project log (section 12)
Repair record (new)	A signed statement by the other party that a matter behind a conduct fact was made right
Escrow state (new)	Where a milestone's money stands with a neutral provider: funded, frozen, released and so on (section 9)
Index (new)	A service that publishes and answers queries over open records (section 14)

Term	Meaning
Profile (new)	A conformance level: pilot, without cryptography, or full (section 4)

4. Project log, envelope and profiles

Every record in a project sits in one append-only log, so all parties compute status from the same history. v0.1 left that history undefined, which let two parties compute different states.

```
{
  "swahast": "0.3",
  "id": "sha256:9f2c...",           // hash of everything below except id
and signatures
  "kind": "agreement",
  "project": "sha256:41ab...",      // the objective's id; omitted for
portable evidence
  "log": { "seq": 42, "prev": "sha256:77e0..." }, // new: position in the
project log
  "authored_by": {
    "principal": "did:key:z6MkOwner...",
    "agent": "did:key:z6MkAgent..." // present only when an agent drafted
the record
  },
  "created_at": "2026-10-05T10:15:00+05:30",
  "derived_from": ["sha256:7c10..."],
  "visibility": "parties",          // public | network | parties | private
  "assumptions": [],               // new: required on agent-authored
records (section 7)
  "body": {},
  "signatures": []                 // appended over time, never part of
the hash
}
```

- **Log order.** Each project record cites the previous entry in `log.prev`. A record citing anything other than the current head is a fork; its author **MUST** re-base it on the head and collect signatures again.
- **Replicas.** Every party keeps a full copy of the log. By default the owner's copy orders new entries.
- **Refused entries.** If a party's valid record is not appended within the period the agreement states, that party **MAY** share it directly with its `prev` reference. The omission becomes a conduct fact against the sequencer (section 12).

- **Portable records.** Evidence and Contribution Records outside any project carry no `log` field.
- **No status field.** Status is computed from the log and signatures (section 6).

Conformance profiles

The rules are the same in both profiles; the cryptography changes who must be trusted, not what the rules are.

Profile	Identity	Consent	Hashing and signatures	Portable outside one app
Pilot	Accounts in one application	A logged action with account and time	Optional	No; records export as JSON
Full	<code>did:key</code> over Ed25519	A signature over the record id	Required: SHA-256 over the RFC 8785 canonical form, Ed25519 signatures	Yes

Both profiles MUST apply every state, authorization, assumption, escrow and conduct rule in this spec. Pilots SHOULD use the pilot profile, so they test coordination rather than machinery.

5. Record kinds

Each lifecycle stage produces its own record kinds. Agents may draft almost all of them, but only principals' consent makes any of them count. Kinds added in v0.2 are marked new.

Stage	Kind	Typical author	Counts when
1 Objective	<code>objective</code>	Owner, often agent-drafted	Owner accepts
2 Capability	<code>requirement</code>	Owner, often agent-drafted	Owner accepts
2 Capability	<code>offer</code>	Contributor or asset controller	Self-signed claim; never binding

Stage	Kind	Typical author	Counts when
3 Evidence	evidence	Subject, source or prior partner	Never binding; carries a verification level
3 Evidence	interpretation	Agent or human reviewer	Never binding; must cite derived_from
4 Formation	formation	Agent or owner	Never binding; must list exclusions
5 Agreement	agreement	Any party, often agent-drafted	Every party accepts the same version
5 Agreement	amendment	Any party or agent	Every party accepts; must cite the version it changes
6 Execution	event	Any party or agent	Never binding
6 Execution	escrow (new)	Owner or escrow provider	Records a milestone's escrow state
7 Outcome	submission	Contributor	Contributor signs
7 Outcome	acceptance	Acceptor, or every party in joint mode	Signed accept or reject
7 Outcome	credit_statement (new)	Every party in joint mode	Every party signs at the credit decision point
7 Outcome	dispute , resolution	Parties; named reviewer	Resolution binds as its tier says (section 12)
8 Memory	closure	Owner, often agent-drafted	Every party signs
8 Memory	contribution_record	Derived from the agreement and acceptance	Its parties sign; in joint mode every party receives one
Any	challenge	The record's subject or a party	Never binding; marks its target contested
Any	explanation (new)	The party a conduct fact concerns	Never binding; travels with that fact
Any	repair (new)	The other party to a conduct fact	Signed statement that the matter was made right

Stage	Kind	Typical author	Counts when
Any	delegation	Principal	Principal signs; ends with a signed revocation

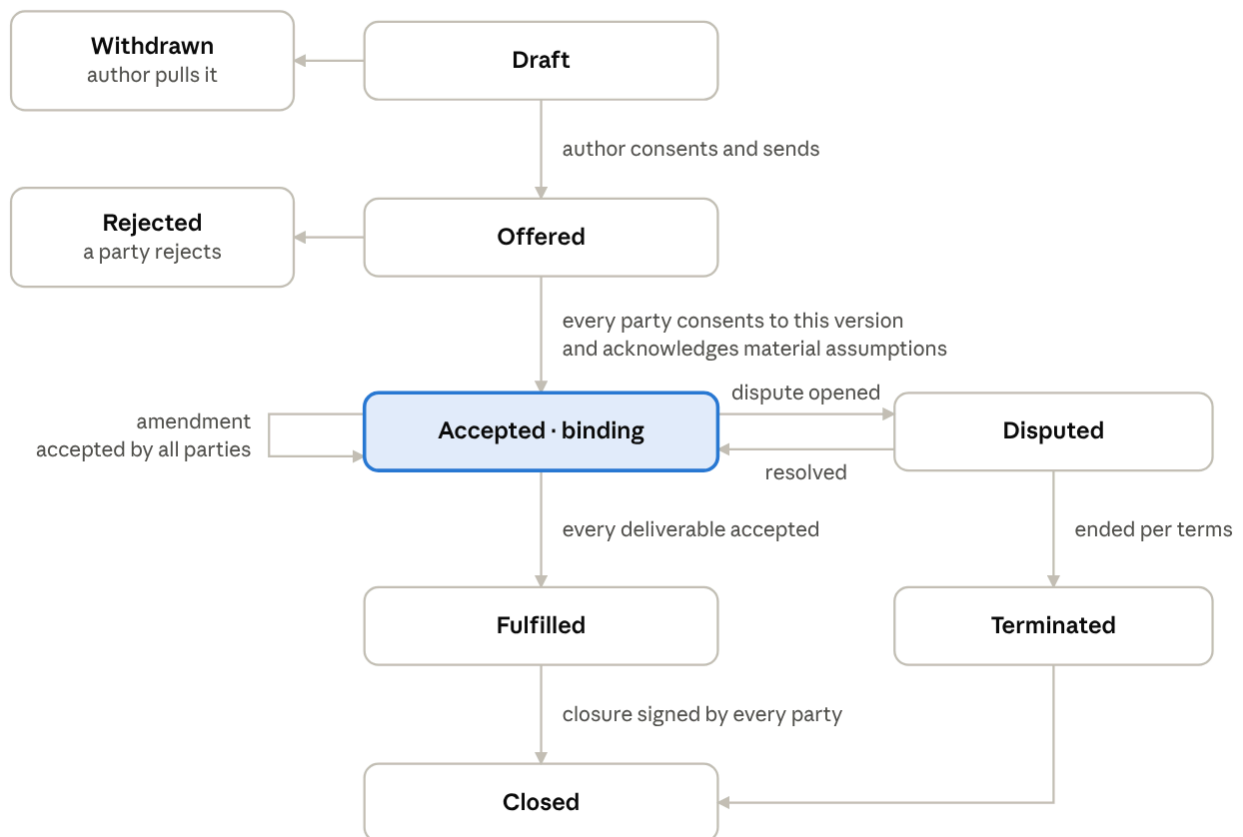
Conduct views are computed from a log, not authored, so they are not a record kind (section 12).

6. Agreement states

An agreement binds only when every party has consented to the same version and acknowledged its material assumptions. Until then it is a draft, however complete it looks.

An agreement binds only when every party consents to the same version

Agreement states in NFP v0.2, supply and joint mode · computed from the project log



Agreement states · binding only at Accepted

Only the highlighted state creates obligations. Disputes and amendments return to it; nothing else enters it.

Status is computed from the project log: a record, its consents and the later records that cite it. Nothing stores it.

Status	Holds when
draft	The record is in the log and its author has not consented
offered	The author has consented
accepted	Every party has consented to this id, acknowledging every material assumption
rejected	A party has rejected this id; a revision is a new record that supersedes it
withdrawn	The author rejected their own record before it was accepted
disputed	An open dispute cites the agreement; related escrow is frozen and windows pause
fulfilled	Every deliverable is accepted: by the acceptor in supply mode, by every party in joint mode
terminated	A resolution or an amendment ended it before fulfilment
closed	A binding closure cites it

Effective terms are the agreement plus its accepted amendments, in log order. A record with an open challenge also shows `contested`; a challenge alone never changes the underlying state. Escrow has its own states, set out in section 9.

7. Authorization and assumptions

Nothing binds a principal without their own consent, and every choice an agent made for them is disclosed before they give it. There is no value below which an agent may accept on a principal's behalf.

Non-delegable acts. Accepting an objective, agreement, amendment, acceptance, credit statement, resolution or closure; opening a challenge or dispute; changing who can see a principal's records. Agents may draft all of these; principals perform them.

Assumptions. An agent-drafted record lists every value it chose that the principal did not state. The signer acknowledges the material ones by id.

```

"assumptions": [
  { "id": "a1", "field": "exit.notice_days", "value": 14,
    "category": "exit", "basis": "No preference stated; template default" },
  { "id": "a2", "field": "scope.deliverables[4]", "value": "Responses to
reviewer comments on method",
    "category": "scope", "basis": "Inferred from the objective's journal
target" }
]
// the principal's accept on that record
{ "by": "did:key:z6MkDevashish...", "act": "accept", "on": "sha256:...",
  "acknowledged": ["a1", "a2"], "sig": "..." }

```

Material categories. Reward, credit, rights, scope, due dates, acceptance criteria, exit, dispute resolution, surviving obligations and visibility, plus any category a principal adds in their delegation. The protocol decides what is material; the agent cannot.

- **A1.** An agent-authored record MUST carry `authored_by.agent`, and its principal MUST hold a `delegation` covering that kind.
- **A2.** Status MUST be derived from the log and consents (section 6). A field claiming acceptance is ignored.
- **A3.** A non-delegable act MUST come from the principal's own key, or own account in the pilot profile. Agent credentials make it invalid.
- **A4.** Consent covers one record id. Any edit produces a new id and voids earlier consents for the edited terms.
- **A5.** An agreement is accepted only when every party consents to the same id.
- **A6.** An `event` never changes state. Later records may cite it.
- **A7.** A validator MUST reject an `interpretation` with an empty `derived_from`.
- **A8.** A `delegation` names the agent, permitted kinds and acts, limits, expiry, the principal's stated preferences and any added material categories. Revocation is a signed record.
- **A9.** An agent-authored record MUST list as an assumption every value in a material category that the principal did not state directly or as a delegation preference.
- **A10.** An `accept` is valid only if its `acknowledged` list contains every material assumption of that record. Other assumptions are accepted with the record.
- **A11.** A material value missing from the assumptions list is grounds for the challenge `undisclosed_assumption` (section 12).
- **A12.** Signing interfaces SHOULD show what changed since the version the signer last saw and highlight terms that differ from template defaults.

Stated preferences shrink the list over time: a principal who has said "exit notice is always 14 days" is never asked about it again. The protocol still cannot prove a human read anything. It guarantees that what they consented to was disclosed, and that an agent which hid something can be shown to have done so.

8. Agreements: supply and joint

An agreement is either supply, where an owner accepts a contributor's deliverable, or joint, where parties contribute, approve together and share credit by a rule fixed in advance. One project can hold both, such as co-authors under a joint agreement and a paid reviewer under a supply agreement.

Both modes encode the paper's primitive, Objective → Capability → Contributor or Asset → Terms → Outcome (§12). v0.1 had only supply mode, which could not hold a co-authorship: neither co-author is the other's client.

Term	Supply mode	Joint mode
Parties	Owner and contributor, or asset controller	Every co-creator, each with roles (CRedit by default)
Scope	Required	Required, with what each party owes and by when
Reward	Money, or an explicit non-monetary return	Structured credit , optionally with money
Acceptance	A named acceptor	Every party approves
Exit	Optional; handled under change terms	Required, tied to the credit threshold
Surviving obligations	Structured list	Structured list
Rights default	retain-preexisting	joint-authorship
Change default	mutual-amendment	mutual-amendment
Resolution default	tiered (section 12)	tiered (section 12)
Escrow	Required for money, per milestone	Required for any money component

Minimum valid agreement. Supply: scope, reward and acceptance. Joint: parties with roles, scope with **owes**, credit, acceptance mode and exit.

```
// body of a joint agreement (illustrative values; the Monsoon to Mandi
pilot)
{
  "mode": "joint",
  "objective": "sha256:m2m-obj...",
  "parties": [
    { "principal": "did:key:z6MkDevashish...",
      "roles": { "conceptualization": "lead", "data_curation": "lead",
"software": "lead", "writing_original_draft": "lead" },
      "owes": [{ "item": "Harmonized price-rainfall panel", "due": "2026-12-
15" }] },
    { "principal": "did:key:z6MkEconomist...",
      "roles": { "methodology": "lead", "formal_analysis": "equal",
"writing_review_editing": "lead" },
      "owes": [{ "item": "Framing memo", "due": "2026-11-15" }, { "item":
"Econometric specification", "due": "2027-01-15" }] }
  ],
  "assets": [
    { "name": "CEDA Agri Market Data", "uri":
"https://agmarknet.ceda.ashoka.edu.in/", "licence": "CEDA-AMD terms: non-
commercial use only", "obligations": ["Cite CEDA-AMD", "Credit CEDA on every
figure that uses it"] }
  ],
  "credit": {
    "type": "co_authorship",
    "venue": { "preprint": "Zenodo", "target": "<journal>", "fallback": ["
<second journal>"] },
    "order_rule": "Owner first if the credit statement lists him as lead on
data_curation, software and writing_original_draft; otherwise order follows
the statement; ties keep the owner first",
    "decided_at": "submission",
    "threshold": "A party who exits before the specification is approved is
acknowledged, not listed as an author"
  },
  "acceptance": { "mode": "all_parties_approve", "criteria": ["Choices
justified in writing", "Results reproducible from raw sources"] },
  "exit": { "notice_days": 14, "credit": "per credit.threshold",
"work_so_far": "Remains usable by the other parties" },
  "survives_closure": [
    { "obligation": "Respond to reviewer comments on method", "by":
"did:key:z6MkEconomist...", "until": "Final decision at the target or fallback
venue" }
  ],
  "rights": { "template": "swahast:rights/0.2/joint-authorship" },
  "change": { "template": "swahast:change/0.1/mutual-amendment" },

```

```
"resolution": { "template": "swahast:resolution/0.2/tiered", "threshold":
"credit disputes and money up to ₹50,000", "reviewer": "did:key:...", "backup":
"did:key:..." }
}
```

- **Roles** use the CRediT contributor roles by default, each with a degree: lead, equal or supporting. An agreement MAY name another role list in role_vocabulary (default: credit) when CRediT does not fit the work, such as a design or software project. The three degrees never change, so credit stays comparable across projects.
- **owes** turns dependencies into commitments of named parties (section 3), so a late input shows as its owner's delay. Each item is a commitment from its party, the debtor, to the other parties, the creditors. It takes effect when the agreement is accepted and is discharged by the record that meets it, late if after the due date in force. An amendment every party signs may release it, and an exit cancels it on the exit terms. An item still unmet at closure is missed. Like every status, these states are computed from the log, never stored.
- **Assets** that sign nothing, such as public datasets, are references with a licence and obligations. A controlled asset, such as a private dataset or a licensed model, still needs a supply agreement with its controller.
- **Rights in joint mode** default to `swahast:rights/0.2/joint-authorship`: (1) each party keeps what it owned before the agreement and licenses it to the project; (2) work made under the agreement belongs to all parties jointly, and no party may publish it alone or assign it, such as to a publisher, without every party's approval; (3) each party may present and cite the work and reuse its own contributions, crediting the others; (4) on publication, code and derived data are released openly, within the source licences of the assets used. Parties may change any of these by explicit term.
- **Templates** are versioned texts, so two parties citing one id agree to the same words. None has been written or reviewed by a lawyer yet.

9. Money: milestone escrow

Paid work uses milestone escrow: each milestone's money is held by a neutral provider before its work starts, and the protocol records what happens to it. The protocol never holds or moves money itself.

Requiring all the money upfront would favour owners who already have capital, the opposite of what the paper wants (§28). Milestone escrow protects the contributor without making the owner lock more than the next stage.

- **E1.** Any money component MUST name milestones, each with an amount and an escrow provider reference, unless escrow is waived.

- **E2.** Escrow can be waived only by an explicit term, `"escrow": { "waived": true, "reason": "..." }`, which counts as a material assumption when an agent drafts it. Typical reasons: revenue share, money held in an institution's grant account, or fees larger than the work.
- **E3.** Work on a milestone **SHOULD NOT** start until its escrow is funded. Whether it was funded in time is a conduct fact.
- **E4.** Each change of state is an `escrow` record citing the provider's reference.

State	Entered when
<code>required</code>	The agreement is accepted
<code>funded</code>	The provider confirms the deposit
<code>frozen</code>	A dispute citing the milestone opens
<code>released</code>	An acceptance of the milestone, or a resolution, instructs release
<code>refunded</code>	A resolution or termination instructs return to the payer
<code>split</code>	A resolution divides the amount
<code>waived</code>	The agreement waived escrow for this milestone

Credit-only agreements, like the first pilot, hold no money; their consequences come from permanent conduct facts (section 12). In India, escrow is generally offered by banks and regulated payment providers with their own fees and rules. Those terms, and how a release instruction reaches the provider, need checking with a lawyer before the first paid pilot.

10. Evidence and formation

Evidence earns weight from who stands behind it, and formation must show what it left out as well as what it proposes. Neither ever produces a score.

Evidence

Each evidence record follows the paper's chain: Subject → Capability claim → Artifact → Source → Role → Provenance → Verification (§10). Verification is a category computed from who has signed:

Level	Who stands behind it
<code>self_claimed</code>	The subject alone, with no artifact
<code>artifact_linked</code>	The subject, pointing to an artifact anyone can inspect
<code>source_confirmed</code>	The artifact's controller attests to the subject's role
<code>counterparty_attested</code>	A partner from a past agreement signs it, usually as a Contribution Record

- A record shows `contested` while a challenge is open and `retracted` once withdrawn. No implementation may publish a combined score from these levels.
- Following `derived_from` from any `interpretation` MUST end at evidence, never only at other interpretations.
- Private evidence stays with its holder; a counterparty can attest to its id without seeing it revealed to others.
- Newcomers have three routes: self-claimed and artifact-linked evidence, a small trial agreement, or vouching by a principal who becomes answerable for it (§16).

Formation

A formation proposal lists configurations that could cover the objective, what each leaves unresolved, and who was excluded and why (§11).

```
// body of a formation record (excerpt)
{
  "configurations": [ { "label": "Earliest start", "assignments": [...],
"unresolved": [...], "tradeoffs": {...}, "conflicts": [...] } ],
  "excluded": [
    { "candidate": "did:key:z6MkC...", "requirement": "sha256:req-eval...",
      "reason": "Unavailable before the due date", "basis":
["sha256:offer-..."] },
    { "candidate": "did:key:z6MkD...", "requirement": "sha256:req-eval...",
      "reason": "Conduct fact: acceptance window exceeded in an earlier
project", "basis": ["sha256:conduct-..."] }
  ]
}
```

- **F1.** No field may order candidates or configurations by quality; clients SHOULD let the owner sort by any trade-off.
- **F2.** Every assignment MUST cite evidence, or say plainly that it has none.

- **F3.** Unresolved requirements and constraint conflicts **MUST** be listed.
- **F4.** A proposal is never accepted. The owner's choice produces agreement drafts, and each candidate accepts or rejects their own.
- **F5.** Proposals are visible to the owner only. A candidate sees the agreement offered to them, not the alternatives.
- **F6.** Every candidate or configuration the proposer considered and dropped **MUST** appear in `excluded` , with a reason and its basis.
- **F7.** A candidate excluded because of a conduct fact **MUST** be listed with that fact, so a human judges it; an agent may never filter on conduct silently.
- **F8.** Proposals **SHOULD** include at least one configuration open to principals with no prior records, where the requirement allows it.

The method that produces a proposal sits outside the protocol. A solver, a language model or a person with a spreadsheet can all emit conforming records.

11. Execution, amendment, credit and closure

During work the protocol records only what may change an obligation. At the end, closure settles the project, and every contributor, the owner included in joint mode, receives a record they keep.

Events

Event (§13)	Usually leads to
<code>milestone_submitted</code>	A <code>submission</code> , then an <code>acceptance</code>
<code>dependency_unavailable</code>	An amendment, or termination under the change terms
<code>scope_change_proposed</code>	An amendment
<code>schedule_changed</code>	An amendment, if it moves a due date
<code>capability_added</code>	A new <code>requirement</code> , then a new agreement
<code>dispute_opened</code>	A <code>dispute</code> record (section 12)

An agent watching a project's tools may file `scope_change_proposed` with `derived_from` pointing at what it saw. The event asks a question; it answers nothing.

Amendments

An amendment names the agreement in `amends` and the exact version it modifies in `base`. `base` MUST be the agreement's current effective version; an amendment built on an outdated version is invalid and must be redrafted. The effective agreement is the original plus its accepted amendments, in log order.

Credit decisions

In joint mode, every party signs a `credit_statement` at the agreement's `decided_at` point. It lists each party's roles with their degree, and the `order_rule` is applied to it. If parties cannot agree on the statement, the dispute goes to the named reviewer (section 12).

Closure

A `closure` record states, for the whole project (§14):

- what was delivered, and which contributions were accepted;
- each milestone's final escrow state;
- the credit outcome, and whether it matched the agreed rule;
- licences and rights still active;
- `survives_closure` obligations, each with a responsible party and an end condition;
- disputes still unresolved;
- what each participant may carry forward.

Closure binds when every party signs. A party silent past the notice period in the change terms does not block it, and the missing signature is recorded.

Contribution Records

A Contribution Record cites the effective agreement, the accepted work, the party's roles and degrees, rights that survive, and any dispute with its resolution. Its parties sign it, and the contributor holds it. In joint mode every party receives one, so the owner's own work enters the network's memory too.

12. Conduct, contestation and disputes

Behaviour is visible and content stays private. Any party may share a project's conduct facts, and the person a fact concerns can always add context or show repair.

Conduct facts

The list is fixed by the spec, so conduct and content cannot be argued case by case.

Conduct fact	Computed from
Obligation met or missed, and days late, measured against the due date in force when it fell due; a date moved by an amendment every party signed before it passed is not lateness	<code>owes</code> and the records that met them
Acceptance decision time against the agreed window	<code>submission</code> and <code>acceptance</code> records
Amendments, and who proposed each	<code>amendment</code> records
Work rejected, then accepted unchanged	Submission ids and acceptances
Escrow funded before the milestone started	<code>escrow</code> records
Exits, and the notice given	Exit events
Disputes opened, their grounds and outcomes	<code>dispute</code> and <code>resolution</code> records
Credit outcome matched the agreed rule	<code>credit_statement</code> and <code>closure</code>
Valid records the sequencer failed to append	Records shared directly with a <code>prev</code> reference

- **C1.** Any party MAY share the conduct view of a project they were in, without the other parties' consent.
- **C2.** The rule is symmetric: it applies equally to owners and contributors.
- **C3.** A conduct view is all or nothing per project, computed deterministically from the log, so a recipient can verify nothing was trimmed.
- **C4.** Amounts, deliverables, terms and messages are content, not conduct, and stay under the agreement's visibility.
- **C5.** Conduct facts are permanent. Expiry would erase good histories along with bad ones.
- **C6.** `explanation` and `repair` records attach to facts and MUST be shared with them. An explanation may give its reason at a general level, such as "personal emergency", without details.
- **C7.** Conduct is shown as a dated sequence across projects, never one project alone, and never as a score.

Challenges

The subject of any record, or a party to it, may file a `challenge` on the grounds `factual_error`, `missing_context`, `unauthorized`, `misattribution` or `undisclosed_assumption`. The target then shows `contested`. Its author may respond, supersede it with a corrected version, or retract it. Nothing is deleted.

Disputes

Every agreement names, at signing, who decides a dispute. The threshold, the reviewer and a backup are material terms.

Tier	Who decides	Appeal
Below the threshold set at signing	A named reviewer, with a backup, both declaring no conflict	Only for a hidden conflict, or a decision contradicting the signed record
Above the threshold	Mediation, then arbitration	As the arbitration terms allow
Credit and authorship	A named reviewer from the same field	As for the lower tier

An open dispute freezes the related escrow and pauses review windows and due dates. A `resolution` records one outcome, `upheld`, `partly_upheld`, `rejected`, `settled` or `terminated`, and instructs any escrow release, refund or split. An agent may summarize a dispute but can never sign its resolution. Whether a named reviewer's decision binds in law, not only in the protocol, may depend on framing it as arbitration; that needs legal review.

13. Research metrics

The project log doubles as the research instrument: seven of the paper's eight pilot metrics (§22) and three new ones come straight from the records. Coordination effort still needs a time diary alongside.

Metric	Computed from
Time to credible formation (§22)	First moment every requirement is covered by an accepted agreement, minus the objective's acceptance time
Capability coverage (§22)	Requirements with no accepted agreement at the project's first event

Metric	Computed from
Coordination effort (§22)	Proxy only: human-authored records and consents per project, paired with participants' time diaries
Agreement amendment rate (§22)	Accepted amendments per agreement, by the term they change
Acceptance rate (§22)	Submissions accepted, divided by submissions made
Dispute incidence (§22)	Disputes per agreement, by grounds, term category and tier
Evidence reuse (§22)	Formation assignments citing a Contribution Record from an earlier project
Repeat formation (§22)	Principals in agreements across two or more projects
Drafting authority (new)	Material assumptions per agent-drafted agreement, and how many principals changed before accepting
Disclosure failures (new)	<code>undisclosed_assumption</code> challenges per agent-drafted record
Escrow reliability (new)	Milestones funded before their start date

Metrics are computed only over records the researcher may see. Consent for research use is a visibility grant the principal signs, not a default.

14. Governance, discovery and prior art

Swahast is an open protocol whose author does not run a platform business on it. Project logs are easy to keep open because the parties hold them; discovery is where power concentrates, so the spec governs it directly.

Discovery

Anyone may run an index over records whose visibility is `public` or `network`. A conforming index:

- **I1.** MUST return records unmodified, with their consents or signatures.
- **I2.** MUST disclose how it orders results.
- **I3.** MUST NOT attach a score, rating or rank to a principal.

- **I4.** MUST return conduct views whole, with their explanations and repair records.

The query interface itself is left to a later version. Because records are open and indexes interchangeable, platforms compete on the quality of their matching and service, not on owning the data (§16, §19).

Stewardship

- **Editor.** The author edits the spec. Every change goes through the public change log, and each version is published on Zenodo with a DOI.
- **Licences.** The spec text under CC BY 4.0, like the whitepaper. The reference implementation under a permissive open-source licence such as Apache 2.0, whose explicit patent grant tells builders the ideas are free to use.
- **Reference implementation.** A minimal open-source application in the pilot profile, built to demonstrate the protocol and to be forked, not to compete.
- **Handover.** Once two independent implementations exist, stewardship moves to a neutral group. The trigger is fixed now, so the promise is checkable.

Interoperability

- **A2A.** Agents exchange Swahast records as task payloads; a project agent can advertise open requirements (§18).
- **MCP.** An MCP server can give a principal's assistant read access and draft-only write access, matching section 7.
- **Verifiable Credentials.** A Contribution Record maps onto a W3C Verifiable Credential: the acceptor issues it, the contributor holds it, and the agreement id is its evidence.
- **Conventional contracts.** A signed agreement's id can be quoted inside an ordinary contract, so both refer to the same terms.

Prior art

The works below were checked against the linked sources between 6 and 8 October 2026. They are the nearest neighbours found in each area, not a complete survey.

Area	Work	What it covers	How Swahast differs
Contracts	Accord Project	Open-source templates for smart legal contracts, drafted by lawyers and executable by software; a Linux Foundation project	Swahast covers the lifecycle around the contract: objective, formation, evidence, consent, conduct and memory

Area	Work	What it covers	How Swahast differs
Contracts	Ricardian contracts (Ian Grigg, 1996)	One document that people and programs can both read, digitally signed and named by a secure identifier	Swahast records follow the same pattern and add a shared project log, computed status and conduct facts
Contracts	Common Paper	Fixed standard terms reviewed once, with negotiation confined to a cover page of variables	Swahast's versioned templates and its highlighting of non-default terms (A12) use the same idea, applied to protocol records rather than contract text
Commitments	Commitment protocols (Singh, 1999, and later work)	Commitments between a debtor and a creditor as first-class objects with a lifecycle, such as detached, discharged, violated and expired, and operations to cancel, release, delegate and assign them	Swahast uses this vocabulary for each party's obligations (section 3). It adds consent by people only, a shared project log and conduct facts; resolving a violation is outside the commitments model
Agent consent	AP2, the Agent Payments Protocol (Google, 2025)	Cryptographically signed mandates prove what a user authorized an agent to buy: an intent mandate sets the rules, a cart mandate approves the exact items and price	Signed user approval of agent actions is not new. Swahast applies it to multi-party agreements and adds assumptions in fixed material categories, each of which must be acknowledged

Area	Work	What it covers	How Swahast differs
Agent consent	Briefing-and-binding envelope (individual Internet-Draft, Morrison and Whiteside, September 2026)	How an agent surfaces a consequential decision and how its principal commits, declines, amends or rejects it	Close to Swahast's agent drafts and non-delegable acts. As read on 6 October 2026, it does not require listing assumptions by material category, and it covers one agent and one principal, not a project's parties
Agent consent	CHEQ (individual Internet-Draft, Rosenberg, White and Jennings, July 2025)	A person confirms an action an agent proposes through a separate confirmation server that signs the decision, so the agent cannot act on its own	Like AP2 and the binding envelope, it covers one action and one person, not a multi-party agreement
Coordination	The Networked Firm (Owocki, Stringer and Ospina, Allo Capital with RnDAO, 2025)	A book on how networks of people and agents form temporary firms and allocate capital: agents negotiate with minimal intervention, code and AI oracles enforce agreements, and reputation becomes scores and tokens	The same setting with the opposite answer on authority: agents draft, only people bind, and conduct is recorded as facts, never scored
Coordination	MPAC (Qian, Fang and Li, arXiv preprint, 2026)	A protocol for agents of different principals working in one session over shared state: a coordinator orders changes, conflicts are objects with each party's position, and escalation goes to an arbiter or a person	MPAC lets agents commit, with human override; here no agent can bind. MPAC covers one session and keeps no reputation; Swahast covers a whole project and its conduct facts, and can run on top of it

Area	Work	What it covers	How Swahast differs
Team formation	Flash organizations (Valentine et al., CHI 2017)	Experts from the crowd assembled on demand into an organization with roles and hierarchy, coordinated on the Foundry platform, for work that needs re-planning	The closest academic precedent for temporary organizations. Swahast is a protocol rather than one platform, and adds consent records, computed status and conduct memory that outlast the project
Team formation	Coral Protocol (2025)	Open infrastructure for AI agents to communicate, coordinate, pay and form trusted teams	Coral forms teams of agents; Swahast forms agreements between human principals, with agents drafting
Credit and reputation	Open Badges 3.0 and W3C Verifiable Credentials 2.0	Portable, verifiable claims of achievement; VC 2.0 became a W3C Recommendation on 15 May 2025	Swahast can issue credentials as an output and records the agreement and acceptance behind them
Credit and reputation	DAO contribution tools, such as SourceCred (no longer maintained)	Contribution graphs that turn tracked activity into scores used to allocate funds	Swahast forbids scores and is built for temporary organizations formed between strangers
Credit and reputation	Colony	Non-transferable reputation, earned mainly by being paid in the colony's token, decaying with a half-life of about 3.5 months	Swahast keeps conduct facts permanently and lets explanation, repair records and chronology outweigh a bad project, instead of letting the past fade

Area	Work	What it covers	How Swahast differs
Credit and reputation	ERC-8004: Trustless Agents (draft, 2025)	On-chain registries for agent identity, reputation and validation; feedback is a submitted numeric value, kept permanently	Swahast's conduct facts are computed from the shared project log, in both directions, never submitted as ratings or turned into scores
Money and disputes	Upwork fixed-price contracts	Milestone funds set aside before work begins, released on approval or automatically after a review period, with a dispute process	Swahast's milestone escrow follows the same pattern as an open protocol: escrow state lives in the parties' log, funding on time is a conduct fact, and the dispute route is fixed at signing
Money and disputes	Kleros (short paper, 2019)	Decentralized arbitration by jurors drawn in proportion to staked tokens, rewarded for voting with the majority, with escalating appeals	Swahast's lower tier uses a named reviewer with a backup and no declared conflict, chosen at signing; above the threshold it uses mediation, then arbitration

Contribution. In the spirit of §26, the claim is narrow. Each part of Swahast has close neighbours above: signed human approval of agent actions, team formation, hash-identified signed records, commitments with a lifecycle and permanent public feedback all exist on their own. What Swahast adds, as of 8 October 2026, is three things: agent-drafted agreements whose assumptions in fixed material categories must each be acknowledged before consent; agreements in which co-creators share structured credit rather than buy and sell work; and permanent, two-way conduct facts computed from a shared project log, with explanations and repair but no scores. Where *The Networked Firm* (Owocki, Stringer and Ospina, 2025) expects agents to negotiate with minimal intervention and contracts to enforce themselves, Swahast keeps people as the only parties who can bind, and makes the agents' work inspectable.

Layering. Swahast governs the agreements and accountability that span a project. Session-level coordination such as MPAC can run underneath: agents may coordinate a working session that way, while Swahast records what the principals agreed, delivered and accepted. The two are complementary, not competing.

15. Scope limits and open questions

v0.2 settled the economic model and the authority line, and v0.3 leaves both unchanged; what remains open is mostly legal and operational, and it can only be answered with real pilots and real advice.

Out of scope for v0.3

- Moving money. Escrow providers hold and move it; Swahast records its state.
- Legal enforceability and choice of law.
- Matching algorithms and agent-to-agent negotiation.
- User interfaces, beyond the signing-screen guidance in A12.
- Tokens or on-chain settlement. A hash-linked log held by the parties needs no blockchain.

Open questions for v0.4

1. **Data protection.** Permanent conduct facts may conflict with erasure rights under data protection laws, such as India's Digital Personal Data Protection Act, 2023. How do permanence and erasure coexist?
2. **Legal weight.** Do signed agreements, and named reviewers' decisions, meet the requirements for electronic contracts and arbitration in the first pilots' jurisdictions?
3. **Escrow in practice.** Which providers can take release instructions from protocol records, and at what cost for small milestones?
4. **Sequencer abuse.** Is direct sharing plus a conduct fact enough to deter an owner who refuses to append a contributor's records?
5. **Discovery interface.** What query interface lets independent indexes interoperate?
6. **Key loss.** In the full profile, how is a lost key recovered without a central custodian?
7. **Collusion.** How are rings of mutual attestation detected without reintroducing scores?
8. **Agent identity.** Should every agent-authored record name the model and version that produced it? Proposed: yes, at least for interpretations and formation proposals.
9. **When to become a firm (§23).** Which pattern in the logs, such as the same principals across several consecutive projects, should prompt the suggestion to form a company?

10. **Upgrading pilot records.** Records made in the pilot profile cannot be verified outside the app that holds them. Can a pilot's closing records, such as the closure and each Contribution Record, be signed in the full profile at the end, so contributors carry verifiable credit even though the earlier history stays unsigned?
11. **Conflict positions.** MPAC records each party's position on a conflict: seen, accepted or disputed. Should a dispute record carry each party's position the same way, before a reviewer is involved?
12. **Declared intent.** MPAC has agents declare an intent, with a scope and an expiry, before they act. Should an agent declare what it is about to draft, and for whom, before the draft exists?

Sources

- Pawar, D. (2026). [The Networked Firm: AI, Coordination, and the Emergence of Temporary Organizations](#), Version 1.0. DOI 10.5281/zenodo.22931577.
- [The Networked Firm concept site](#).
- [CEDA Agri Market Data](#), Ashoka University.
- Owocki, K., Stringer, D. and Ospina, D. (2025). [The Networked Firm: Capital Allocation in the Age of Blockchain and AI](#). Allo Capital with RnDAO.
- Qian, K., Fang, X. and Li, Z. (2026). [MPAC: A Multi-Principal Agent Coordination Protocol for Interoperable Multi-Agent Collaboration](#). arXiv:2604.09744.
- Singh, M. P. (1999). [An ontology for commitments in multiagent systems: Toward a unification of normative concepts](#). Artificial Intelligence and Law, 7, 97–113. The lifecycle states named in section 14 follow Singh's [course slides on commitments](#) (NC State University, 2019).
- Rosenberg, J., White, P. and Jennings, C. (2025). [CHEQ: A Protocol for Confirmation AI Agent Decisions with Human in the Loop](#). Individual Internet-Draft.
- Standards and works named here (RFC 2119, RFC 8785, did:key, Ed25519, CRediT, A2A, MCP, Apache 2.0, the Digital Personal Data Protection Act) are cited from general knowledge and should be checked against their current versions. The works compared in section 14 were checked against the sources linked there between 6 and 8 October 2026.