

Contribution Value System Conceptual Engineering Plan

Contribution Value System Engineering Design Plan v1.1.2

Document role: A companion to the whitepaper for technical teams

Reference: Contribution Value System Whitepaper v1.8.2-R2, Controlled Revision of the Exit-Closure Interface |

Permanent asset: P05-CV-002 | Engineering version: v1.1.2, Withdrawal of Responsibilities and Retirement of Overreaching Interfaces

September 2026

Author: Zijun Fu | Produced by Civilization Leap Research Group

About this document

This document accompanies the engineering implementation of the Contribution Value System Whitepaper v1.8.2-R2. It addresses technical teams and answers how to record, verify, review, appeal, audit, and migrate. The whitepaper governs institutional definitions and boundaries; this plan governs data structures, state machines, factual-explanation interfaces, event ledgers, access control, testing, and migration. Engineering implementation must not rewrite the whitepaper in reverse or directly generate governance sovereignty, voting rights, resource eligibility, professional qualifications, execution privileges, or personality hierarchies from Contribution Value.

v1.1.2 synchronization scope: Carry forward v1.1.1's T-10 separation of facts, eligibility, and rights; retire governance weighting, governance voting, and resource allocation responsibilities from the Contribution Value system; replace overreaching modules with current relevance views, external factual explanations, resource-request facts, and authorization/decision receipts; synchronize the whitepaper v1.8.2-R2 boundaries between exit, responsibility, and closure; and retain responsibilities for contributing subjects, P0-P4, community attribution, conservation of totals, Resource Value facts, appeals, event ledgers, auditing, and migration.

C1-R2 controlled revision boundaries

Retired: Governance-weight formulas, CV-weighted voting, proposal voting, resource-application ranking, resource-allocation execution, and their APIs/tests. If deployed, old interfaces must be archived read-only and return an explicit retired status. They must neither continue accepting writes nor be repackaged as current authority.

Retained: Contribution facts, evidence, assessments, subjects and membership relations, P0-P4 dynamic confirmation, internal community attribution and conservation of totals, Resource Value facts, external requests, factual explanations, objections, appeals, event ledgers, auditing, and migration.

Responsibility boundary: Governance V3.0 or other legitimate real-world structures determine public purposes and procedures; the Resource Pool/Shared World carry resource facts and legitimate use-right relationships; the Contribution Value system provides only sourced, explainable, contestable factual material with an expiry and records external decision/authorization receipts.

Parameter boundary: Capacity, concurrency, and infrastructure numbers in this draft may serve only as synthetic load-testing examples. They do not establish real pilot participant counts, durations, weights, priorities, voting rules, resource-allocation rules, or automated-decision parameters.

Notation conventions follow.

- [TBD]: Parameters or implementation details that need not be settled at this stage; to be decided after pilot data, ethical review, or community governance.
- [Reference value]: Provisional values for engineering tests or capacity planning, not institutional constants.
- [Architecture recommendation]: Technical implementation recommendations that do not change the whitepaper's institutional boundaries.

1 User experience goals from a recording tool to an echo of meaning

Baseline product experience for the Contribution Value System:

- Users should not feel scored, monitored, or subjected to moral demands.
- Users should feel that genuine contributions are being recorded safely, with little burden, and with the option to exit.
- Contribution records must not create status hierarchies, social credit, or illusions of economic incentives.
- The value of contribution records is in being seen, rather than exchanging them for rewards.

The product experience should not make users feel scored, monitored, or subjected to moral demands. It should make them feel that genuine contributions can be recorded safely, with little burden, and with the option to exit.

Startup-stage experience goal: "My genuine contributions can be recorded safely." Avoid a sense of scoring, monitoring, and moral pressure.

Transition-stage experience goal: "Other contributors and I are forming a trustworthy collaboration network." Avoid comparisons, performative behavior, and hierarchies.

Stable-stage experience goal: "Contribution records are supporting broader public meaning and long-term civilizational goals." Avoid platformization, self-enclosure, and worship of tools.

System design must therefore always serve four principles: low-burden recording, protection of dignity, stage-specific explainability, and auditable correction.

v1.1.2 current experience and rights requirements:

- P0-P2 stage-specific values must be explicitly labeled "estimated / initially confirmed / stage-validated." They must not be presented to users as permanent final values.
- The community's overall contribution and internal member subdivisions must separately display their sources, totals, and attribution rules, preventing organizational accounts from being misrepresented as the person in charge's personal account.
- Display historical contribution records and current relevance views in separate columns. Civilization can remember historical contributions; current relevance can describe only a record's factual relationship to a specific question. It does not automatically establish resource priority, governance influence, voting rights, or eligibility.

- Neither user-facing nor backend systems may establish psychological scoring fields for “purity of motivation,” “sincerity,” “degree of altruism,” “sense of meaning,” “obedience,” or the like.

Part One System architecture

C1-R2 controlled revision note: This document carries forward B5 W1 and v1.1.1's T-10 boundaries and withdraws responsibilities in protocols, schemas, interfaces, roles, and tests. All retired interfaces are handled through explicit migration records. This revision creates no new real-world authority, freezes no real-world parameters, and does not change the whitepaper's institutional substance.

T-10 separation of facts interpretation eligibility and rights EMB-3 and EMB-4

Four-layer separation: Factual records answer “What happened?” Contribution interpretation answers “What consequences followed?” Eligibility judgments answer “Are the conditions for a limited role met?” Basic rights and general political eligibility stand independently. None of these four layers may automatically imply another.

Against a total ledger of meaning: Contribution Value records concrete actions and consequences after finite subjects enter reality; it does not measure personality value, life meaning, purity of goodwill, or subject rank. Meaning is the root, and its sources are plural. Being unrecorded, not contributing, or temporarily unable to contribute is no reason to diminish subsistence, dignity, care, or basic rights.

AGI and communities: AGI may assist recording, verification, explanation, and auditing, but computational ability does not grant contribution attribution, ownership of meaning, or governance sovereignty. Community contributions must not swallow individuals' actual responsibilities. Organizational dissolution, member objections, and attribution disputes must be retained.

Evidence and dynamic confirmation: P0–P4 describe the maturity of contribution facts and confirmation of consequences, not personality or subjectivity ranks. An upgrade changes only a record's evidential status and permitted uses. It does not automatically generate resource, governance, or legal eligibility.

Authority and objections: Every resource application, governance participation, professional role, or execution task requires a separate jurisdictional, institutional, eligibility, and current-authorization basis. The Contribution Value system may provide only factual explanations and relevance materials, while preserving explanation, objection, appeal, correction, exit, and human responsibility. It must not itself rank, vote, allocate, or authorize. Example parameters, test proportions, and code samples are not real-world implementation parameters.

Exit and data: After exit, records cease entering new current relevance explanations, external factual requests, or any ranking/authority use. Existing snapshots expire according to their authorized scope and validity period. Necessary preservation of historical evidence is separate from current access rights. Deletion, de-identification, audit retention, and relinking each depend on purpose, duration, jurisdiction, and current authorization.

Sources and non-effect: Based on whitepaper v1.8.2-R2 and the C1 audit, v1.1.2 retires overreaching schemas, interfaces, roles, and tests or withdraws their responsibilities; it creates no new real-world authority. Sources: Contribution Value System Whitepaper v1.8.2-R2; T-10 Consolidated Volume of Interdisciplinary Theoretical Upgrades V1.0; T-10 Master Register of Derived Fields for Six Types of System Components by Object, Version, and Stage B3 V1.0; T-10 Master Embedded-Audit Register for Six Types of System-Component Document Families by File, Version, and Stage B4 V1.0; WP5-BATCH-C C1 Institutional Core Audit V1.0.

1.1 Overall architecture diagram textual description

The source diagram's layers and components are reproduced below in reading order.

User Layer

Web client | Mobile app | Open API | Offline entry point

↓

Gateway Layer

Authentication | Rate limiting and circuit breaking | Request routing | Log auditing

↓

Core Business Layer

Contribution-record module | Assessment module | Relevance-view module | Factual-explanation module

Resource-request facts and receipts module | Objection/appeal module | Notification/push module

↓ Continued on the next source page

Architecture diagram continued



Supporting Services

Anti-cheating / integrity assurance | Privacy protection service | Audit logs | Rule references and versions

Evaluator matching service | Domain configuration service | External request/receipt service | Time/relevance engine



Data Layer

Primary data (PostgreSQL) | Cache layer (Redis) | File storage (MinIO/S3) | Search index (Elasticsearch)



Optional blockchain/DID layer [TBD]

1.2 Core modules

1.2.1 Contribution record module

Responsibility: Manage the complete lifecycle of contribution records.

Core functions: Submit contribution declarations, supporting individuals, communities, organizations, and other contributing subjects and multiple contribution types; preliminary review, including format checks and classification; contribution status management, keeping declaration status separate from P0–P4 maturity; querying and searching contributions; editing and withdrawing contributions in specified states.

Key design point: Time spent and frequency of contribution declarations are real-deployment parameters. This draft requires only low burden, withdrawability, and explainability; it freezes no uniform number of minutes or monthly submissions.

Support evidence import from digital collaboration platforms within explicit authorization and the minimum necessary scope to reduce repeated declaration burdens. The actual deployment entity configures average monthly declarations, rate limits, and anomaly thresholds according to risk, burden, and compliance requirements. This master draft freezes no uniform numbers.

1.2.2 Assessment module

Responsibility: Coordinate multiple evaluators' quality assessments of contribution facts and stage-specific results, without judging subjects' inner motives.

Core functions: Random evaluator matching excluding interested parties; distribution of multidimensional assessments of authenticity, effectiveness, scarcity, and synergy; aggregation and calculation of assessment results; triggering and routing assessment disputes.

Key design points: Evaluator composition and numbers must be derived from the real scenario according to contribution type, scope of impact, difficulty of evidence, risk, uncertainty, and degree of dispute. "High CV" or downstream resource/governance uses must not automatically confer greater power. Evaluators must exclude interested parties and retain entry points for declining tasks, objections, and review.

- Evaluator reliability is used only for quality calibration and retrospective review of the current assessment task and does not create personal credit. Consistency-monitoring thresholds are configured after scenario- and risk-based validation.

1.2.3 Current relevance views and factual explanation module

Responsibility: Create explainable current relevance views and factual-explanation snapshots based on contribution facts, evidence maturity, domain relevance, responsibilities undertaken, timeliness, and uncertainty; do not calculate governance weights, voting rights, resource eligibility, or decision priorities.

Core functions: Separate historical contributions from current relevance; aggregate factual sources and evidence chains; explain domain/time/responsibility relationships; register uncertainties and counterexamples; bind purpose, authorization, validity period, and version; generate factual-explanation snapshots.

Key design point: Outputs must consist of "facts + basis + applicable scope + unknowns + a dispute entry point." They must not output overall subject scores, rankings, governance weights, resource-allocation sequences, or automatic eligibility.

Contribution Value has zero influence on basic subsistence, life safety, basic medical care, personal dignity, basic freedom, exit, appeals, and similar matters. Any external use remains the responsibility of real-world entities with the relevant authority and duties, who bear decisions and consequences.

1.2.4 Resource request facts and authorization receipt interface module

Responsibility: Record the purposes, subjects, domains, scope of impact, and factual bases of external resource requests; generate explanations of contribution facts; and receive authorization, refusal, deferral, withdrawal, and usage-result receipts produced by the Resource Pool, Shared World, or legitimate real-world procedures.

Core functions: Register requests; extract the minimum necessary facts; explain histories of resource inputs, maintenance, and repair; receive external authorization-source receipts; write back actual usage and outcome events; record disputes, withdrawals, and closure.

Key design point: The Contribution Value system does not maintain a list of available resource pools, calculate application priority, execute resource allocation, or establish use rights. Contribution records can be limited reference facts only within an explicit jurisdiction and specific purpose. The corresponding rights holders, legitimate public rules, or professional procedures determine resource eligibility, priority under scarcity, and actual use. The system preserves only traceable requests, factual explanations, and external-decision receipts.

1.2.5 Anti cheating and record integrity assurance module

Responsibility: Detect and handle anomalous behavior to maintain system trustworthiness.

Core functions: Detect false declarations through evidence verification and time/location plausibility; circular reciprocity through transaction-graph analysis; collusive assessments through score similarity and network relationships; split declarations through behavior-pattern analysis; identity forgery [TBD]; and random spot-check verification.

Responses: Address only record integrity and current role permissions. Actual jurisdictions/rules determine specific periods, reviewing subjects, and procedures.

- False declarations: Revoke or mark false records. Where necessary, impose temporary, appealable restrictions on the declaration interface with explicit scope and duration.
- Circular reciprocity: Revoke anomalous records. If assessment tasks are involved, evaluator reliability may be calibrated for that task/domain, without creating a personal credit score.
- Collusive assessments: Suspend the relevant assessment-task eligibility and initiate independent review. No uniform period may be preset.
- Split declarations: Consolidate calculations and prompt correction.
- Identity forgery: Revoke the relevant records or operational permissions and initiate independent review. Actual jurisdictions and organizational rules determine the reviewing entity and accountability procedures.

1.2.6 Privacy protection module

Responsibility: Balance transparency and privacy.

Core functions: Protect beneficiary information through anonymization; tiered visibility of contribution records; manage authorization for sensitive operations; de-identify data; manage informed consent.

Key design points: Beneficiaries may choose anonymity; beneficiaries must not be compelled to express public gratitude; images of vulnerable people must not be exchanged for Contribution Value; contribution records are private by default, with selective display at the individual's choice.

1.2.7 Contributing subjects and membership relations module

Responsibility: Maintain an independent entity layer for “who the contributing subject is,” supporting natural persons, temporary collaborative groups, project communities, long-term communities, organizations, and cross-organizational collaborations.

Core functions: Create/dissolve subjects; membership relations and authorization; representatives as authorized operators only, not contribution owners; separation of community and individual accounts; subject lifecycle and historical archive management.

Key design point: Being a community's legal representative, administrator, initiator, or actual controller does not automatically confer personal contribution attribution.

1.2.8 Dynamic confirmation and maturity state machine module

Responsibility: Implement P0–P4 value maturity, observation periods, stage reviews, dynamic adjustment, stable confirmation, and long-term retrospective additions.

Core functions: Maturity state machine; schedule observation points; append outcome evidence; version valuations; preserve adjustment reasons and before/after values; P3 stable confirmation; P4 appends only newly manifested value without rewriting established history.

Key design point: P0 is only an estimate and does not enter formal accounts; P1–P2 must be explicitly labeled stage-specific; P3 establishes stable historical records; P4 enters the ledger as a new addition event.

1.2.9 Overall confirmation internal subdivision and conservation of totals module

Responsibility: Support member subdivisions, community-retained portions, and external-collaborator attribution after a community acquires Contribution Value as a whole, and verify that the same result is not counted twice.

Core functions: Versions of attribution plans; prior agreements; member confirmation; attribution-total validation; freezing disputed portions; independently confirming attribution of subsequent P4 increments.

Key design point: For the same contribution event, “community-retained + member-attributed + external-collaborator-attributed” must equal that event's currently confirmed attributable total. Creating sub-records must not multiply counts.

1.2.10 Resource Value and Resource Pool interface module

Responsibility: Distinguish resource base value from resource realized value after actual use, while retaining ongoing regenerative value, maintenance costs, and future option value.

Core functions: Resource registration; availability status; usage events; maintenance responsibilities; stage reviews of base/realized values; necessary redundancy and safety-reserve flags; explicit separation of Resource Value and Contribution Value.

Key design point: Possessing resources is not contributing. Corresponding contribution records may arise only after resources are genuinely opened, actually used, responsibilities undertaken, and verifiable results produced.

1.2.11 Review appeal and audit interface module

Responsibility: Provide unified review and appeal entry points for initial values, stage adjustments, internal community attribution, Resource Value, current relevance views, external factual explanations, requests and receipts, and record-integrity responses.

Core functions: Register objections; supplement evidence; independent review; suspend the effect of disputed portions; version rulings; audit the whole chain.

Key design point: Adjustments may occur, but “silent ledger edits” are forbidden. Any key change affecting subject records, current relevance views, external factual explanations, or receipt status must be explainable, traceable, and appealable. The Contribution Value system must not use a review interface to rewrite external decisions.

1.3 Data flows between modules

v1.1.2 end-to-end data flow:

Subject/community submits contribution facts or resource-usage results

↓

[Contributing subjects module] Confirms subject boundaries, membership relations, and representative authority

↓

[Contribution record module] Creates ContributionEvent + P0/P1 records

- Format/evidence-completeness checks
- Duplicate-declaration and conflict-of-interest prechecks
- Prohibits writing motive-purity/personality-psychology fields

↓

[Assessment module] Assesses authenticity, effectiveness, scarcity, synergy, and domain conditions

↓

[Dynamic confirmation module] P1 initial value → P2 stage validation → P3 stable confirmation

- Every adjustment creates a non-overwritable valuation version and event log
- Realized historical contributions and current relevance views are stored separately
- Future newly manifested value is handled as P4 addition events

↓ Two parallel branches

[Community subdivision module]: Overall acquisition; internal subdivision; community-retained portion; conservation-of-total checks

[Resource Value module]: Base value; realized value; ongoing value/maintenance; future option value

↓ Branches rejoin; continued on the next source page

↓

[Current relevance view] → Project collaboration / resource-request explanations / governance factual explanations (no eligibility, priority, or power created)

↓

[Audit and appeals] Preserve a complete trail of changes to subjects, valuations, attribution, Resource Value, factual explanations, requests, and receipts.

Engineering discipline: All subsequent review changes current views through “new event + new version,” without physically overwriting historically confirmed records.

Part Two Data model

2.1 Core entity definitions

2.1.1 Participant

```
interface Participant {
  // Identity
  id: string; // UUID, globally unique
  decentralized_id?: string; // DID [Architecture recommendation, TBD]
  // Basic information
  display_name: string; // Display name; may be anonymous
  created_at: timestamp;
  status: 'active' | 'review_limited' | 'withdrawn' | 'archived';
  // User labels: participating normally / declaration or assessment
  // permissions temporarily limited / exited / historical archive
  // Historical summaries and current relevance views:
  // calculated fields, not accounts, balances, or power
  current_effective_relevance?: RelevanceSnapshotRef;
  // Current contextual relevance view; must include purpose,
  // domain, basis, and validity period
  // Frontend shows contribution records by type and maturity,
  // never overall subject scores or rankings
  thought_contribution_record_summary?: Decimal;
  behavior_contribution_record_summary?: Decimal;
  system_contribution_record_summary?: Decimal;
  collaboration_contribution_record_summary?: Decimal;
  // Reputation information
  evaluator_reliability: Decimal;
  // Assessment-task quality calibration only; not personal credit
  // or a personality score
  lastactivityat: timestamp;
  // Participation information
  join_domain: string[]; // Participating domains
  role: 'contributor' | 'evaluator' | 'review_coordinator' |
    'auditor' | 'system_admin';
  // Technical operational roles, not governance status or public power
  // Privacy settings
  privacy_settings: {
    profile_visible: boolean; // Profile visibility
```

```

    contributions_public: boolean; // Contribution-record publicity
    realidentitylinkable: boolean; // Real-identity linkability
  };
  // Exit information
  withdrawn_at?: timestamp;
  withdrawal_status?: 'pending' | 'completed';
}

```

2.1.2 ContributionSubject

```

interface ContributionSubject {
  id: string;
  subject_type: 'person' | 'temporary_group' | 'project_community' |
    'long_term_community' | 'organization' | 'cross_org_collective';
  display_name: string;
  status: 'active' | 'closing' | 'dissolved' | 'archived';
  created_at: timestamp;
  dissolved_at?: timestamp;
  representative_actor_ids: string[];
  // Operational authorization only, not personal ownership
  default_domain_ids?: string[];
  operating_rules_ref?: string;
  // Community internal rules/charter reference; a source only,
  // not established by CVS
  privacy_settings: JSONB;
}

```

2.1.3 SubjectMembership

```

interface SubjectMembership {
  id: string;
  subject_id: string; // FK -> ContributionSubject
  participant_id: string; // FK -> Participant
  relation: 'member' | 'coordinator' | 'authorized_representative' |
    'external_collaborator';
  joined_at: timestamp;
  left_at?: timestamp;
  authorization_scope?: JSONB;
  // Permitted operations; no presumption of contribution ownership
  attribution_consent_required: boolean;
}

```

2.1.4 Contribution

```

interface Contribution {
  id: string;
  event_id: string; // FK -> ContributionEvent;
  // stable ID of one real contribution event
  subject_id: string; // FK -> ContributionSubject;
  // no longer presumed to be a natural person
}

```

```

declared_by_actor_id: string;
// Natural person/authorized operator actually initiating declaration
domain_id: string;
type: 'I' | 'B' | 'S' | 'C';
subtype: string;
title: string;
description: text;
flags: { nondirectbenefit: boolean; high_cost: boolean;
  intergenerational: boolean; innovative: boolean; };
evidence: EvidenceRef[];
beneficiary?: { count_min: number; count_max: number;
  is_anonymous: boolean; consent_given: boolean; };
declared_at: timestamp;
occurred_at?: timestamp;
workflow_status: 'draft' | 'pending_review' | 'under_evaluation' |
  'under_observation' | 'disputed' | 'closed';
maturity_stage: 'P0' | 'P1' | 'P2' | 'P3' | 'P4';
observation_plan?: { next_review_at?: timestamp; review_window?: string;
  expected_result_signals?: string[]; };
current_valuation_id?: string; // Latest valuation version
stable_history_cv: Decimal;
// Cumulative historical contributions stably confirmed at P3;
// a record, not an account balance or power
current_relevance_snapshot_id?: string;
// Relevance-explanation snapshot for the current question;
// generates no eligibility or ranking
attribution_plan_id?: string;
// Internal attribution subdivision of overall community contribution
dispute_ids?: string[];
integrity_flags: { evidence_verified: boolean;
  pattern_check_passed: boolean; manual_review_required: boolean; };
}

```

Prohibited fields: motive_score, altruism_score, sincerity_score, meaning_score, obedience_score, or any other quantitative fields for “purity of motivation / personality psychology” must not enter the contribution-record master data model.

2.1.5 ContributionValuation

```

interface ContributionValuation {
  id: string; contribution_id: string; revision_no: number;
  maturity_stage: 'P0' | 'P1' | 'P2' | 'P3' | 'P4';
  value_before?: Decimal; value_after: Decimal; delta: Decimal;
  basis: { contribution_facts?: string[]; result_evidence?: EvidenceRef[];
    responsibility?: string[]; risk?: string[]; externalities?: string[]; };
  decision: 'estimate' | 'maintain' | 'increase' | 'decrease' |
    'stabilize' | 'reopen' | 'long_term_addition';
  reason: text; reviewer_ids: string[]; created_at: timestamp;
  appealable: boolean; supersedes_id?: string;
}

```

```
}

```

2.1.6 CollectiveAttributionPlan

```
interface CollectiveAttributionPlan {
  id: string; contribution_id: string; version: number;
  attributable_cv: Decimal; community_retained_cv: Decimal;
  attributions: { subject_id: string; cv: Decimal; basis?: text;
    consent_status?: string; }[];
  external_attributions?: { subject_id: string; cv: Decimal; basis?: text; }[];
  conservation_check: { expected_total: Decimal; actual_total: Decimal;
    passed: boolean; };
  status: 'draft' | 'member_review' | 'confirmed' | 'disputed' | 'superseded';
  confirmed_at?: timestamp;
}
```

Conservation-of-total check: $\text{community_retained_cv} + \sum \text{attributions.cv} + \sum \text{external_attributions.cv} = \text{attributable_cv}$.

Any inequality prevents the attribution plan from taking effect.

2.1.7 ResourceValueRecord

```
interface ResourceValueRecord {
  id: string; resource_id: string; owner_subject_id: string;
  maturity_stage: 'P0' | 'P1' | 'P2' | 'P3' | 'P4';
  base_components: { capacity?: Decimal; scarcity?: Decimal;
    lifetime?: Decimal; location?: string; maintenance_cost?: Decimal;
    substitutability?: Decimal; };
  realized_components?: { actual_usage?: Decimal;
    outcome_evidence?: EvidenceRef[]; maintenance_burden?: Decimal;
    positive_externality?: Decimal; negative_externality?: Decimal; };
  regeneration_value?: Decimal;
  future_option_value?: Decimal;
  // Analytical field, not a tradable asset price
  current_context_value?: Decimal;
  // Nonlinear contextual assessment result; parameters await pilots
  next_review_at?: timestamp; revision_no: number;
}
```

2.1.8 ContributionEvent ledger

```
interface ContributionEvent {
  id: string; contribution_id: string;
  event_type: 'declared' | 'subject_changed' | 'valuation_created' |
    'stage_advanced' | 'attribution_changed' | 'appeal_opened' |
    'appeal_resolved' | 'resource_fact_linked' | 'fact_brief_issued' |
    'external_receipt_recorded' | 'reopened' | 'long_term_added';
  actor_id: string; occurred_at: timestamp;
  payload_hash: string; previous_event_hash?: string;
  details: JSONB; // Append-only events; history is not overwritten
}
```

```
}

```

2.1.9 Assessment

```
interface Assessment {
  // Identity
  id: string;
  contribution_id: string; // FK -> Contribution
  evaluator_id: string; // FK -> Participant
  // Assessment information
  round: number;
  assigned_at: timestamp;
  submitted_at?: timestamp;
  // Scores
  scores: {
    authenticity: number; // 1-5 points
    effectiveness: number; // 1-5 points
    scarcity: number; // 1-5 points
    synergy: number; // 1-5 points
  };
  // Evaluator comments
  comment?: text;
  // Evaluator declaration of position
  conflict_of_interest: boolean;
  // Whether a conflict of interest exists; judgments of motivation
  // must not be declared as an assessment basis
  conflict_explanation?: text;
  // Status
  status: 'assigned' | 'completed' | 'skipped' | 'conflict_flagged';
  // Evaluator reliability calibrates assessment quality only;
  // it does not establish general personal evaluator reliability
  // Timeliness tracking
  deadline: timestamp;
}
```

2.1.10 Domain

```
interface Domain {
  id: string;
  name: string;
  description: text;
  parent_id?: string; // Parent domain; supports hierarchy
  // Contribution-type assessment configuration, not governance weights;
  // derived from actual scenarios
  assessment_profile: {
    I: Decimal;
    B: Decimal;
    S: Decimal;
    C: Decimal;
  };
}
```

Source page 15

```

};
// Domain-specific rules
rules: {
  min_evaluators: number; // Minimum evaluator count
  evidence_requirements: string[];
  specialflagsenabled: string[];
};
// Rule and review context: external references only;
// carries no governance authority
review_context: {
  external_rule_ref?: string;
  last_review_at?: timestamp;
  next_review_at?: timestamp;
};
// Status
status: 'active' | 'under_review' | 'archived';
// Configurability flag
local_adjustable: boolean; // Whether local adjustment is permitted
adjustment_range?: { // Permitted adjustment range [TBD]
  min: Decimal;
  max: Decimal;
};
}

```

2.1.11 External factual requests and decision receipts ExternalReferenceRequest

```

interface ExternalReferenceRequest {
  id: string;
  request_type: 'governance_fact_brief' | 'resource_fact_brief' |
    'professional_role_fact_brief';
  requesting_actor_id: string;
  requesting_system: string;
  purpose: text;
  domain_id?: string;
  subject_ids?: string[];
  contribution_ids?: string[];
  requested_fields: string[];
  legal_or_rule_basis_ref: string;
  authorization_ref: string;
  affected_subject_ids?: string[];
  requested_at: timestamp;
  expires_at?: timestamp;
}

```

Source page 16

```

status: 'draft' | 'authorization_verified' | 'issued' | 'disputed' |
'expired' | 'closed';
fact_brief?: {
  snapshot_id: string;
  facts: FactRef[];
  current_relevance_context?: RelevanceContext[];
  evidence_refs: EvidenceRef[];
  uncertainties: string[];
  excluded_fields: string[];
  generated_at: timestamp;
  non_authoritative: true;
};
external_decision_receipt?: {
  external_process_id: string;
  decision_type: 'approved' | 'rejected' | 'deferred' | 'withdrawn';
  decision_ref: string;
  authority_source_ref: string;
  received_at: timestamp;
};
dispute_ids?: string[];
audit_event_id: string;
}

```

Engineering boundary: CVS only generates factual-explanation snapshots and records external decision/authorization receipts. It does not create proposals, organize voting, calculate voting rights, approve resources, or replace real-world decision-making.

2.2 Relationship diagram ER diagram description

Core relationships v1.1.2:

Participant —< SubjectMembership >— ContributionSubject

ContributionSubject 1:N Contribution

Contribution 1:N Assessment

Contribution 0:1 CollectiveAttributionPlan

Contribution 1:N ContributionValuation

Continued on the next source page.

Contribution 1:N ContributionEvent (append-only)

Domain 1:N Contribution

ResourceValueRecord N:1 ContributionSubject; resource-usage outcomes may serve as evidence in a Contribution stage review.

ExternalReferenceRequest references Contribution/ContributionSubject and generates a non-authoritative FactBrief. External decisions/authorizations are written back only as Receipts.

Relationship discipline:

- Separate “declaring operator” from “contributing subject”: a representative/participant may act on a community’s behalf but cannot thereby acquire its contributions.
- Separate “contribution events” from “internal attribution”: member attribution references the same contribution event rather than copying it.
- Separate “historical records” from “current relevance”: P3/P4 history may be retained long term; current relevance is formed only with explicit purpose, domain, authorization, and validity period.
- Separate “Resource Value” from “Contribution Value”: changing resource capacity does not automatically mean changing a subject’s contribution. Real responsibilities and results must establish the causal connection.

2.3 Key fields

Field	Type	Description	Design decision
current_effective_relevance	JSONB/Ref	Current effective relevance view	Not a balance, total score, or power; must include source and expiry
workflow_status	Enum	Contribution workflow status	Separate from P0–P4 maturity
timestamp	timestamp	All timestamps use UTC	Convert to local time for client display
evidence	JSONB	Evidence stored as JSONB	Flexible support for multiple evidence types
flags	JSONB	Contribution flags use JSONB	Easy extension with new flag types
privacy_settings	JSONB	Privacy settings use JSONB	Flexible visibility configuration
subject_id	UUID	Subject to whom the contribution genuinely belongs	Not the declaring operator’s ID

Field	Type	Description	Design decision
maturity_stage	Enum	P0-P4 value maturity	Managed separately from workflow status
stable_history_cv	Decimal	Cumulative P3/P4 stable historical records	Not an account balance or permanent power
current_relevance_snapshot	UUID/JSONB	Relevance-explanation snapshot for the current question	Factual explanation only; no eligibility, voting rights, or priority
valuation_revision	Integer	Valuation version number	Append a new version for each adjustment; never overwrite old values
attribution_plan	JSONB/Entity	Community internal attribution subdivision plan	Must pass conservation-of-total checks
future_option_value	Decimal/Index	Analytical field for resources' future option value	Not an asset price or direct CV conversion item

Part Three API design overview

3.1 Core interfaces REST format

Contribution declaration interfaces

Method	Path	Description
POST	/api/v1/contributions	Submit a new contribution declaration
GET	/api/v1/contributions/:id	Get contribution details
GET	/api/v1/contributions	Query contribution list with filtering
PATCH	/api/v1/contributions/:id	Edit a contribution in draft state only
DELETE	/api/v1/contributions/:id	Withdraw a contribution awaiting assessment only
POST	/api/v1/contributions/:id/dispute	Object to a contribution result

Assessment interfaces

Method	Path	Description
GET	/api/v1/assessments/pending	Get pending assessment tasks
POST	/api/v1/assessments	Submit assessment results
GET	/api/v1/assessments/:id	Get assessment details

Method	Path	Description
POST	/api/v1/assessments/:id/decline	Decline an assessment task

External factual-explanation interfaces, carrying neither voting nor governance weights

Method	Path	Description
POST	/api/v1/reference-requests	Create a factual request with purpose, authorization, and expiry
GET	/api/v1/reference-requests/:id	Get request, authorization-verification, and dispute status
GET	/api/v1/fact-briefs/:snapshot_id	Get a factual-explanation snapshot that cannot be silently rewritten
POST	/api/v1/reference-requests/:id/dispute	Object to factual scope, omissions, basis, or use
POST	/api/v1/external-decision-receipts	Record a decision receipt from an external governance/professional procedure

Retired interfaces: /api/v1/proposals, /api/v1/votes, and /api/v1/governance/weight* cease accepting new writes. Where historical deployments exist, return 410 Gone or a read-only archival explanation and refer callers to the external governance system.

Resource-request facts and authorization-receipt interfaces, making no allocation decisions

Method	Path	Description
POST	/api/v1/resource-requests	Register external resource-request facts and purposes
GET	/api/v1/resource-requests/:id	Query factual explanations and external authorization/refusal/withdrawal receipts
POST	/api/v1/resource-use-receipts	Record actual usage, responsibilities, and outcome facts following authorization

Retired interfaces: /api/v1/resources/pools, /applications, and /allocation* are no longer provided by CVS. P05-RESOURCE-003 carries resource facts; the Shared World/contracts/legitimate procedures establish effective use rights; clearly responsible subjects undertake actual execution.

Participant interfaces

Method	Path	Description
GET	/api/v1/participants/me	Get current user information
PATCH	/api/v1/participants/me/privacy	Update privacy settings
POST	/api/v1/participants/me/withdraw	Apply to exit
GET	/api/v1/participants/:id/profile	Get a participant's public profile

Domain interfaces

Method	Path	Description
GET	/api/v1/domains	Get the domain list
GET	/api/v1/domains/:id	Get domain details

Contributing-subject interfaces

Method	Path	Description
POST	/api/v1/subjects	Create a contributing subject; non-individual subjects require an explicit type
GET	/api/v1/subjects/:id	Get subject details and a public rules summary
POST	/api/v1/subjects/:id/memberships	Add/update membership relations and operational authorization
POST	/api/v1/subjects/:id/dissolve	End a temporary/project community and enter the winding-up process

Dynamic-confirmation interfaces

Method	Path	Description
GET	/api/v1/contributions/:id/valuations	Get P0-P4 valuation history
POST	/api/v1/contributions/:id/reviews	Initiate stage review/long-term additions
POST	/api/v1/contributions/:id/appeals	Appeal a valuation, attribution, or adjustment

Community-attribution interfaces

Method	Path	Description
GET	/api/v1/contributions/:id/attribution	Get the overall acquisition and internal attribution plan
PUT	/api/v1/contributions/:id/attribution	Submit/revise an attribution plan; conservation of totals is mandatory
POST	/api/v1/contributions/:id/attribution/confirm	Member confirmation or objection

Resource Value interfaces

Method	Path	Description
POST	/api/v1/resource-values	Register resource base value and observation plan
GET	/api/v1/resource-values/:id/history	Get P0-P4 version history of Resource Value
POST	/api/v1/resource-values/:id/reviews	Conduct stage review based on actual usage/results

3.2 Interface parameter explanations brief

POST /api/v1/contributions

Request body v1.1.2:

```
{
  "subject_id": "uuid (required)",
  "declared_by_actor_id": "uuid (server/session resolved)",
  "domain_id": "uuid (required)",
```

Source page 21

```

"type": "I | B | S | C (required)",
"subtype": "string (required)",
"title": "string (required, max 200)",
"description": "string (required, max 5000)",
"flags": { "nondirectbenefit": false, "high_cost": false,
  "intergenerational": false, "innovative": false },
"evidence": [ ... ],
"beneficiary": { "count_min": 1, "count_max": 10,
  "is_anonymous": true, "consent_given": true },
"occurred_at": "ISO8601 timestamp (optional)",
"collective_context": {
  "is_collective_result": true,
  "attribution_rule_ref": "string (optional)",
  "community_retained_ratio": "optional, not authoritative until attribution review"
},
"observation_plan": { "next_review_at": "optional",
  "expected_result_signals": ["..."] }
}

```

Response:

```

{ "id": "uuid", "event_id": "uuid", "workflow_status": "pending_review",
  "maturity_stage": "P0", "created_at": "ISO8601" }

```

The server must reject or ignore any prohibited psychological-scoring fields, such as motive_score / altruism_score / sincerity_score / meaning_score / obedience_score, and record this in input-validation logs.

Compatibility note [Architecture recommendation]: If v1.0.3/v1.1.x interfaces have been deployed, migrate subject_id, P0–P4, and attribution fields through /v2 or a compatibility adapter. If not yet deployed, the v1.1.2 data contract may be used directly as the first production schema.

POST /api/v1/assessments

Request body:

```

{
  "contribution_id": "uuid (required)",
  "scores": {
    "authenticity": 1-5,
    "effectiveness": 1-5,
    "scarcity": 1-5,
    "synergy": 1-5
  },
  "comment": "string (optional, max 2000)",
  "conflictofinterest": false,
  "conflict_explanation": "string (optional)"
}

```

POST /api/v1/reference-requests

Request body:

```
{
  "request_type": "governance_fact_brief | resource_fact_brief | professional_role_fact_brief",
  "purpose": "string (required)",
  "legal_or_rule_basis_ref": "string (required)",
  "authorization_ref": "string (required)",
  "contribution_ids": ["uuid"],
  "requested_fields": ["facts", "evidence", "current_relevance", "uncertainties"],
  "affected_subject_ids": ["uuid"],
  "expires_at": "ISO8601 timestamp"
}
```

Response:

```
{
  "request_id": "uuid",
  "fact_brief_snapshot_id": "uuid",
  "excluded_fields": ["basic_rights", "vote_weight", "resource_eligibility", "personality_score"],
  "non_authoritative": true,
  "dispute_url": "/api/v1/reference-requests/:id/dispute"
}
```

3.3 Database migration and compatibility strategy

v1.1.2 migration principle: Do not delete data history; stop new writes for overreaching responsibilities. Preserve old records as read-only evidence, without interpreting them as current authority.

If v1.0.3/v1.1.0/v1.1.1 has not entered production: Create the database directly with the v1.1.2 schema, avoiding retention of Participant = contributing subject, governance weights, CV-weighted voting, or resource-allocation modules.

If an old version is deployed: First dual-write/backfill ContributionSubject, P0–P4, attribution, and event chains; then freeze writes to old fields; finally verify, object by object, historical contribution counts, subject attribution, event chains, factual snapshots, and appeal entry points.

Field migration: Stop new writes to cv_balance, current_effective_weight, and governance_weight. Map them to read-only legacy fields or current_effective_relevance snapshots. All relevance views must carry purpose, domain, basis, time, and validity period.

Governance-interface retirement: Archive GovernanceProposal, votes, governance/weight, and related roles/tests read-only. The external governance system itself takes responsibility for proposals, participation, voting, decisions, execution, and correction. CVS only generates factual explanations and records external receipts.

Resource-interface retirement: CVS no longer provides resources/pools, applications, or allocation. Historical applications may migrate to ExternalReferenceRequest. Preserve historical allocation results only as external_decision_receipt, explicitly identifying the authorization source at the time. They cannot renew automatically.

Attribution-terminology migration: Migrate CollectiveAllocationPlan and /api/v1/contributions/:id/allocation to CollectiveAttributionPlan and /attribution. This structure handles only internal attribution and conservation of totals for the same contribution event; it does not concern resource allocation.

Part Four Privacy and security

4.1 Identity protection plan

Design goal: Protect personal identity while ensuring record trustworthiness.

Plan:

1. Identity layers

- System identity: A unique identifier used internally, not exposed externally.
- Display identity: A user-defined display name; may be a pseudonym or anonymous.
- Verification identity: Real-name information used in anti-cheating verification, stored separately from system identity.

2. Separate identity storage

User table (PII isolation zone): System ID; verification identity, encrypted; contact details; audit fields.

Public user table (public zone): System ID; display name; public privacy settings; public contribution-record references.

1. Optional DID integration [Architecture recommendation, TBD]

- Participants may choose to associate a decentralized identity (DID).
- DID is not mandatory; not linking it does not affect eligibility to participate.

4.2 Data minimization principle

Implementation measures follow.

Principle	Implementation
Collect only necessary data	Limit declaration-form fields; prohibit collecting information unrelated to contributions
Limit retention periods	Retain contribution records permanently; delete temporary assessment data after 90 days [TBD]
Limit access scope	Role-based access control (RBAC) + attribute-based access control (ABAC)
Privacy by default	Contribution records are private by default; users actively choose publicity

Detailed data-retention rules

Data type	Recommended retention rule
Contribution-record summaries	May be retained long term
Raw evidence materials	Retain for 1–3 years under specified periods, or delete/archive after disputes end
Beneficiaries' personal information	Minimize storage; preferably do not store
Assessment-process data	May be retained during pilots; delete or de-identify according to audit cycles in formal operation
Publicly displayed data	Must be de-identified
Exited participants' records	No longer used for current weights; retain only de-identified historical archives

4.3 Anonymizing contribution records

Tiered anonymity strategy

Scenario	Anonymity level	Description
Public contribution wall	High	Show only type, domain, and approximate time
Within-community display	Medium	Show display name and contribution type
Personal profile	Low	Users decide what to display
Arbitration review	Traceable	Full information may be queried after authorization

Technical implementation: The contribution-wall API returns aggregate data, not details linkable to individuals. User privacy settings govern profile content. Audit logs record all sensitive-data access.

4.4 Access control strategy

Multilayer access control: diagram continued on the next source page.

Access control architecture

Layer One: Authentication

- Basic authentication: Email/mobile number + password
- Enhanced authentication: Multifactor authentication (MFA) [Recommended]
- Optional authentication: DID/blockchain signatures [Architecture recommendation, TBD]

Layer Two: Role permissions (RBAC)

- contributor: Declare contributions and view own records
- evaluator: Declare contributions + assigned assessment tasks
- review_coordinator: Assessment-task routing + dispute coordination; no governance authority
- auditor: Auditing, evidence access, and correction verification; no routine decision-making authority
- system_admin: Technical operations; must not rewrite facts, rules, or external decisions

Layer Three: Attribute permissions (ABAC)

- Domain binding: Evaluators may assess contributions only in their own domains
- Interested-party exclusion: Automatically exclude parties with conflicts of interest
- Time limits: Sensitive operations must be completed within specified periods
- Rate/scope limits: Prevent interface abuse, without establishing resource eligibility or subject ranks

Layer Four: Audit tracking

- Log all sensitive operations for auditing

Beneficiary protection: Use a separate encryption key for beneficiary information. Ordinary administrators cannot access beneficiary information. Beneficiaries may withdraw informed consent at any time.

4.5 Audit log design

Scope of logging

Category	Recorded content	Retention rules
Authentication	Login time, IP, authentication method	Configure by jurisdiction and security purpose
Contribution declaration	Declarant, time, content summary	Configure by purpose, disputes, and historical responsibility
Assessment operations	Evaluator, scores, conflict declarations	Configure by assessment review and audit needs
Factual explanations	Request purpose, authorization, field scope, snapshot, expiry	Expiry of current use does not delete necessary historical evidence
Administrative operations	Technical administrator operations, reasons, affected objects	Must be traceable; no silent ledger changes
Data access	Sensitive-data queries and exports	Configure by jurisdiction, purpose, and minimum necessary period

Category	Recorded content	Retention rules
System events	Exceptions, circuit breaking, configuration changes	Configure by recovery, disputes, and security responsibilities
Valuation events	P0–P4 stage changes, before/after values, delta, basis, reviewer	Long term
Attribution events	Community subdivisions, member confirmation, conservation-of-total results	Long term
Resource Value events	Version changes to base/realized/future option values	Long term as required for auditing
Appeal events	Objections, evidence supplements, rulings, rollback/reopening	Long term

Log format:

```
{
  "timestamp": "ISO8601",
  "eventtype": "contributionsubmitted",
  "actor_id": "uuid (system ID)",
  "actor_role": "contributor",
  "target_id": "uuid",
  "target_type": "contribution",
  "ip_address": "string (encrypted storage)",
  "user_agent": "string",
  "details": {
    "domain": "uuid",
    "type": "B",
    "flagsapplied": ["nondirect_benefit"]
  },
  "consent_verified": true
}
```

Log access control: Ordinary administrators may view only logs related to their own operations. Independent audit roles may access logs within legitimate authorization and the minimum necessary scope. No single administrator, operator, evaluator, or external requester may monopolize evidence, rule interpretation, or termination entry points. Key ledger logs cannot be modified or deleted in place; only new events may be appended. Corrections are represented through supersedes/reopen relationships.

Log archiving, retention, and deletion are jointly determined by jurisdiction, purpose, dispute status, responsibility, and risk. No uniform period is fixed.

4.6 Motivation neutrality and anti manipulation data constraints

The engineering system handles only verifiable facts, responsibilities, risks, and results. It does not establish internal scores for “who is more sincere, more altruistic, or more meaningful.”

Fields/features prohibited from entering the CV calculation chain: purity of motivation, psychological loyalty, personality rank, obedience, probability of being persuaded, and exploitability of vulnerabilities.

Research data permitted only with isolation: Participation friction, reasons for exit, voluntary feedback, and similar data may improve product experience, but must be separated from contribution valuation and current relevance services by database/permissions. Without new explicit authorization, they must not be written back to contribution records, factual snapshots, or any downstream eligibility.

If AGI/model outputs contain “system inferences,” they must be explicitly labeled as inferences and provide their principal bases and uncertainty. A system understanding participants better cannot justify expanding its decision-making power.

Part Five Extensibility design

5.1 Supporting multiple communities

Architecture design: Multi-community support

Community A	Community B	Community C
Independent domains	Shared domains	Independent domains
Independent rules	Cross-domain rules	Independent rules
Local rules	Mutual-recognition rules	Local rules

All three connect to a shared foundational layer:

- Cross-domain contribution circulation
- Contribution Value mutual-recognition protocol
- Unified authentication
- Independent dispute routing

Implementation points:

1. Community isolation

- Each community has an independent database schema or database instance.
- Domain definitions are independent within each community, though it may choose a standard domain set.

- Contribution-recording, assessment, and review rules within communities are configurable, but must obey the whitepaper’s shared baselines and real-world jurisdictions.

2. Cross-community collaboration

- Shared domains: Domains jointly maintained by multiple communities.
- Cross-domain contributions: Contributors may declare that contributions benefit other communities.
- Contribution circulation: Implemented through mutual-recognition protocols; see Appendix H.

3. Data isolation strategies

```
-- Option A: Independent schemas, recommended for small deployments
community_a.contributions
community_b.contributions
-- Option B: Shared schema + community ID field,
-- recommended for medium deployments
contributions WHERE community_id = 'A'
-- Option C: Independent database instances, for large deployments
communityadb.contributions
communitybdb.contributions
```

5.2 Supporting cross community contribution circulation

V1.1.0 addition: Cross-community mutual recognition must distinguish both “participant identity” and “contributing-subject identity.” Temporary communities or cross-organizational collaborations may exist across communities, but each community may read only the minimum subject information and stable historical summaries needed for mutual recognition.

A community’s overall contribution established in Community A must not automatically be split into individual members’ contributions on entering Community B. Cross-domain/cross-community subdivision must reference the original event and confirmed attribution plan to prevent duplicate measurement.

Circulation scenarios

Scenario	Description	Implementation
Cross-domain contributions	Contributions in Community A affect Community B	Inter-community mutual-recognition protocol
Bridging contributions	Collaboration connecting different communities	Bridging flag + centrality cap
Resource-fact coordination	Cross-community resource-request facts and results	Resource Pool/Shared World/external legitimate procedure interfaces

Technical implementation:

1. Contribution export

{

```
"export_id": "uuid",
"originalcontributionid": "uuid",
"exportingcommunity": "communityA",
"receivingcommunity": "communityB",
"contribution_hash": "sha256",
"signature": "communityAsigning_key",
"exported_at": "ISO8601",
"expires_at": "ISO8601"
}
```

1. Contribution import verification

- Verify the source community's signature.
- Check compliance with minimum mutual-recognition standards.
- Check hash consistency.
- Create a local linked record.

2. Mutual-recognition protocol; see whitepaper Appendix H

- Both parties recognize global baselines.
- Neither party has major violation records.
- Record formats are compatible.
- Dispute-handling mechanisms are consistent.

5.3 Supporting future feature extensions

Extension design principles

Principle	Implementation
Extensible domains	Hierarchical domain structure, with dynamic additions
Extensible contribution types	Namespaced type codes, such as I.5 for scientific discovery
Extensible assessment dimensions	Configurable dimensions; adding dimensions does not affect historical data
Extensible flag types	Contribution flags use JSONB and permit new flags
Hot-updatable rules	Business rules use a rule engine supporting dynamic updates

Extension-point design:

```
// Example extension point: contribution-type registry
interface ContributionTypeRegistry {
  register(type: ContributionType): void;
  get(typeCode: string): ContributionType;
```

```

    listByCategory(category: string): ContributionType[];
  }
  // Example extension point: assessment-dimension engine
  interface AssessmentDimensionEngine {
    registerDimension(dimension: AssessmentDimension): void;
    calculateScore(contributionId: string,
      dimensions: string[]): Promise<Score>;
  }
  // Example extension point: current relevance context registry;
  // not a uniform decay/weight formula
  interface RelevanceContextRegistry {
    register(domainId: string, contextRuleRef: ExternalRuleRef): void;
    createSnapshot(subjectId: string, purpose: string,
      domainId?: string): Promise<RelevanceSnapshot>;
  }

```

5.3.1 P0 to P4 state machine extension principles

States and functions:

- **P0 estimate:** Records only facts awaiting verification and observation plans. Enters neither stable contribution records nor any downstream eligibility, priority, voting rights, or authorization.
- **P1 initial confirmation:** Establishes a stage-specific initial value explicitly labeled “initial.” It may enter factual explanations with explicit authorization but must retain uncertainty and must not generate resource/governance eligibility.
- **P2 stage validation:** Stage-specific results exist. Values may be added, maintained, or reduced, or observation extended. Clear maturity labels remain.
- **P3 stable confirmation:** Main results have manifested; the record enters stable historical contribution records. Current relevance must still be separately generated according to purpose, domain, responsibility, timeliness, and evidence.
- **P4 long-term addition:** When genuine long-term value previously unknowable emerges, append a new delta event without writing back to or overwriting the original P3 history.

State transitions are implemented through appended ContributionValuation and ContributionEvent versions, not by deleting old values. Major errors, fraud, or new evidence may trigger reopen, but the original version and reopening reasons must remain.

5.4 Offline support plan overview

Goal: Support basic functions in weak-network environments.

Tiered design

Functional layer	Offline support	Description
Contribution declaration	Supported	Local drafts + queued synchronization

Functional layer	Offline support	Description
Assessment tasks	Partial	View downloaded tasks; temporarily save scores
Contribution queries	Partial	View local cache; synchronize periodically
Current relevance views	Unsupported	Requires online reading of latest facts, purpose, authorization, and timeliness
Factual-explanation requests	Unsupported	Requires online verification of authorization, validity, and minimum necessary scope
Appeal/objection drafts	Supported	Save in a local queue; user confirms submission after reconnecting

Technical plan [TBD; detailed design to be refined on the basis of Appendix H]:

1. Local storage

- IndexedDB stores contribution drafts and cached data.
- Encrypt sensitive information at rest.

2. Offline queue

- Service Worker intercepts requests.
- Write to a queue while offline.
- Synchronize automatically after reconnecting.

3. Conflict handling

- Detect conflicts using timestamps + hashes.
- Prompt users to resolve conflicts manually.
- Use eventual consistency for key operations.

4. Synchronization strategy

- Periodic full synchronization, once daily.
- Real-time incremental synchronization when a network is available.
- Manually triggered synchronization initiated by users.

5.x Wording principles for contribution confirmation notifications

Notifications should emphasize “Your contribution has been recorded,” “Thank you for taking responsibility,” and “This record will enter your contribution archive.” Avoid excessive gamification, rankings, and stimulating language.

Recommended: “Your contribution has been recorded. Thank you for leaving this trace of care / protection / collaboration in the community.”

Not recommended: “Congratulations on earning XX points! Your ranking has risen by XX places!”

The system may display contribution records, confirmation stages, bases, and current relevance explanations, but must not make a single number, ranking, rank, or implied power the primary motivation for participation.

5.x Strengthening data retention principles

Data-retention strategies must obey “record contributions, do not possess people.” Long-term retention should primarily concern de-identified contribution summaries, statistical results, records necessary for auditing, and historical archives, rather than continued retention of identifiable beneficiary information, raw images, or private personal materials.

Principles for exited participants’ records: 1. Stop including them in new current relevance explanations, external requests, or any ranking/authority use. 2. They may be used for retrospectives, statistical analysis, or dispute auditing when de-identified and supported by a legitimate purpose. 3. Relinking to personal identity requires new authorization from that individual or explicit, reviewable audit conditions. 4. Raw evidence should be deleted or archived according to retention periods, not exposed indefinitely.

5.x Product antipatterns

Any product design that turns contribution records into score competition, status comparison, moral check-ins, or display of beneficiaries violates the Contribution Value System’s original purpose. The product antipatterns are as follows.

Explicitly avoid these design directions:

1. Ranking-centered interfaces: Lists, ranks, and rankings as the homepage’s core display.
2. Check-in altruism: Encouraging users to manufacture contributions for the sake of maintaining consecutive records.
3. Turning beneficiaries into exhibits: Using images of older people, children, or vulnerable groups as contribution evidence or publicity material.
4. Highly stimulating notifications: Gamified language such as “Congratulations on leveling up” or “You overtook XX people.”
5. Contribution details public by default: Disclosing specific contribution recipients, locations, times, and beneficiary information without users actively choosing this.
6. Organizations absorbing individual contributions: By default counting members’ contributions in organizational totals without retaining records of individuals’ actual responsibilities.
7. High-CV privilege entry points: Presenting high-CV users in the interface as ranked identities or a quasi-privileged class.
8. Weak appeal entry points: Hiding appeal, correction, and exit options too deeply.

Product design must avoid distorting “contribution records” into “score competition.”

Technical fields and user display standards

Core principle: Databases record facts, evidence, valuation versions, and contextual relevance; interfaces present records, bases, maturity, and objection entry points.

Technical fields may exist internally, but must conform to their institutional meaning. Their “existence in the backend” cannot justify hidden governance weights, resource rankings, or personality scores. The following mappings are mandatory.

Technical field	Database/backend	Recommended user-interface display
current_effective_relevance	Current effective relevance view including purpose, domain, basis, and timeliness	Current relevance explanation; not a balance or ranking
thought_contribution_record_summary	Confirmed intellectual-contribution summary	Intellectual contribution records
behavior_contribution_record_summary	Confirmed behavioral-contribution summary	Behavioral contribution records
system_contribution_record_summary	Confirmed systemic-contribution summary	Systemic contribution records
collaboration_contribution_record_summary	Confirmed collaborative-contribution summary	Collaborative contribution records
confirmed_contribution_value	Current confirmed value/valuation version	Magnitude and maturity of contribution records
fact_brief_snapshot_id	External factual-explanation snapshot	Factual-reference explanation; creates no voting rights or resource eligibility
evaluator_reliability_context	Assessment-task quality calibration	Assessment-quality explanation; not personal credit

Terms prohibited in user interfaces:

- Prohibited: Score / ranking / rank / points / balance / redemption / circulation / high-scoring user / VIP / ranking system.
- Permitted expressions: Record, trajectory, archive, contribution summary, maturity, current relevance explanation, factual request, appeal, and correction.

System design must distinguish technical fields from user perception.

Databases may preserve confirmed values by type, valuation versions, time-adjustment bases, and relevance contexts, but must not generate overall subject rankings, ranks, benefit redemption, voting weights, resource eligibility, or social credit. Users should feel they are leaving contribution records that can be corrected, not being scored by a system.

v1.1.2 current display mappings:

- maturity_stage: Display “estimated / initially confirmed / stage-validated / stably confirmed / long-term addition,” never “rank.”

- **stable_history_cv**: Display “stably confirmed historical contribution records.” Do not merge with current relevance or any power into a total score.
- **current_effective_relevance**: Display “current relevance explanation” only under explicit purpose and authorization, with basis, scope, timeliness, and uncertainty. Do not use for rankings, voting rights, or resource priority.
- **CollectiveAttributionPlan**: Community members may see attribution bases relevant to themselves and conservation-of-total results. Transparency requirements must not disclose other members’ private information they lack permission to view.
- **ResourceValueRecord**: Distinguish “resource base capacity” from “value released after actual use.” Do not display resource market prices directly as CV.

Part Six Technology selection recommendations

Note: The following are architecture recommendations, not requirements. Technology choices should weigh team familiarity, deployment environment, and business scale.

6.1 Database selection considerations

Type	Recommended scenario	Alternatives or explanation
Relational databases		
PostgreSQL	Primary data storage; first choice	Core data such as contribution records, users, and assessments
MySQL	Lightweight deployments	Small pilots
Time-series databases		
TimescaleDB	CV decay calculations and trend analysis	Optional; a PostgreSQL extension may also suffice
Document databases		
MongoDB	Semistructured data and log storage	Noncore data
TimescaleDB	Event time series, observation windows, and relevance-context analysis	Optional; must not embed uniform decay or weight formulas
pgvector	Contribution-similarity matching	Architecture recommendation, TBD
Cache		
Redis	Sessions, caching, real-time queues	Required
Search		
Elasticsearch	Full-text search and log analysis	Optional; PostgreSQL full-text search can meet initial needs

Recommended plan (100-person pilot):

Single PostgreSQL instance

- Primary data tables, such as contributions and participants

Synthetic capacity baseline A (example: 100 test accounts, not an actual pilot participant count):

- Redis (cache + Session)
- Local filesystem (evidence-file storage)

Expansion plan (1,000+ people):

PostgreSQL primary-replica replication

- Read-only replicas for query separation

Synthetic capacity baseline B (example: 1,000 test accounts, not a real-world stage):

- MinIO/S3 (file storage)
- Elasticsearch (search + logs)

6.2 Authentication plan

Approach	Applicable scenario	Description
Self-built authentication		
Email/mobile number + password	Basic scenarios	Salted password hashes; MFA support
Social login		
OAuth 2.0 + major platforms	Reduce registration barriers	Account-linking issues must be handled
Decentralized identity		
DID/W3C Verifiable Credentials	Long-term vision	Architecture recommendation, long-term goal

Recommended plan, pilot stage: Email + strong password (bcrypt/scrypt); optional TOTP two-factor authentication; JWT session tokens, short-lived, + Redis blacklist.

6.3 Analysis of blockchain and DID use cases

Recommended plan, controlled test environment: Email + strong password (bcrypt/scrypt).

Use case	Suitability	Description
Tamper-resistant logs	Medium	Audit logs and contribution-record hashes
Decentralized identity	Medium	DID control of personal identity and contribution records
Cross-community mutual recognition	Medium	Trust anchor for contribution circulation
Real-time transactions	Low	CVS is not a trading system; blockchain is unsuitable
High-frequency operations	Low	High on-chain gas costs

Recommended strategy follows.

Stage	Recommendation	Description
Pilot stage, 100 people	Do not introduce blockchain	Traditional database + audit logs
Expansion stage, 1,000+ people	Optional	Contribution-record hash archiving only
Mature stage	Introduce as needed	Decide according to community needs
Synthetic test A, example 100 accounts	Do not introduce blockchain	Traditional database + append-only audit logs
Synthetic test B, example 1,000 accounts	Optional	Event-hash archive verification only

Component	Recommended technology	Description
Backend framework	Node.js/TypeScript or Go	Balance rapid iteration against performance
Frontend framework	React/Vue + TypeScript	Prioritize team familiarity
Mobile	React Native or Flutter	Prioritize cross-platform support
API gateway	Kong or Nginx	Rate limiting, authentication, routing
Containerization	Docker + Docker Compose	Required for development/test environments
Orchestration	Kubernetes	Production environment, 1,000+ people
CI/CD	GitHub Actions / GitLab CI	Automated build and deployment
Monitoring	Prometheus + Grafana	Basic monitoring required
Orchestration	Kubernetes	Candidate for large-scale deployment; requires capacity testing

Part Seven Testing strategy

7.1 Unit testing strategy

Test coverage priorities

Module	Test focus	Coverage target
Contribution records	State transitions and validation rules	90%+
Assessment calculation	Evidence vectors, layered confirmation, and consistency of reasons	95% of critical paths
Relevance views	Purpose/domain/timeliness/basis and expiry handling	95% of critical paths
External references	Request authorization, minimum facts, receipts, and disputes	100% of authority boundaries
Privacy protection	De-identification and access control	90%+

Module	Test focus	Coverage target
Subject relations	Community membership/representative authorization, dissolution and winding-up	90%+
Maturity states	P0-P4 transitions, reopen, P4 additions	95%+
Conservation of totals	No double-counting between overall acquisition and internal attribution	100% of critical paths
Event ledger	Append-only logs and version-chain integrity	95%+
Motivation neutrality	Prohibit psychological fields in valuation chains	100% input validation

Test examples:

```

describe('Contribution state transitions', () => {
  test('Enters pending_review after declaration', () => {
    const contribution = Contribution.create({...});
    expect(contribution.status).toBe('pending_review');
  });
  test('Enters confirmed after assessment is completed', () => {
    const contribution = Contribution.create({...});
    contribution.submitForEvaluation();
    contribution.completeAssessment({scores: {...}, final_cv: 10});
    expect(contribution.status).toBe('confirmed');
  });
  test('Evaluators with conflicts of interest are excluded', () => {
    const evaluator = createParticipant({role: 'contributor'});
    const contribution = createContribution({contributor_id: evaluator.id});
    const matched = AssessmentMatcher.findEvaluators(contribution);
    expect(matched).not.toContainEqual(evaluator);
  });
});
describe('Community conservation of totals and P0-P4', () => {
  test('Overall attribution cannot exceed the currently confirmed attributable total', () => {
    const plan = createAttributionPlan({
      attributable_cv: 100, community_retained_cv: 20,
      member_attributions: [30, 50]
    });
    expect(validateConservation(plan)).toBe(true);
  });
  test('P4 long-term additions create a new delta rather than overwrite P3', () => {
    const p3 = createValuation({stage: 'P3', value_after: 100});
    const p4 = addLongTermValue(p3, 25);
    expect(p4.delta).toBe(25);
    expect(loadValuationHistory().some(v => v.id === p3.id)).toBe(true);
  });
  test('Motive-purity fields cannot enter contribution declarations', () => {

```

```
    expect(() => submitContribution({motive_score: 0.9})).toThrow();
  });
});
// The former governance-weight-calculation test suite is retired.
// These negative tests verify that CVS does not generate voting rights,
// resource eligibility, or decision-making power.
describe('Retirement of overreaching interfaces and factual explanations', () => {
  test('CVS must not generate voting rights or resource eligibility from CV', () => {
    const brief = createFactBrief({ contribution_id,
      purpose: 'governance_review' });
    expect(brief.vote_weight).toBeUndefined();
    expect(brief.resource_eligibility).toBeUndefined();
    expect(brief.non_authoritative).toBe(true);
  });
  test('Factual requests without purpose or authorization are rejected', () => {
    expect(() => createReferenceRequest({ contribution_id })).toThrow();
  });
  test('External decision receipts must record the source of authority and must not write back to CV', () => {
    const receipt = recordExternalDecisionReceipt({ authority_source_ref,
      decision_ref });
    expect(receipt.authority_source_ref).toBeDefined();
    expect(loadContribution(contribution_id).stable_history_cv).toBe(original_cv);
  });
  test('Retired interfaces return an explicit status', async () => {
    await expect(call('/api/v1/governance/weight'))
      .rejects.toMatchObject({ status: 410 });
    await expect(call('/api/v1/votes')).rejects.toMatchObject({ status: 410 });
    await expect(call('/api/v1/resources/allocation/x'))
      .rejects.toMatchObject({ status: 410 });
  });
});
```

7.2 Integration testing priorities

Test scenarios

Scenario	Test content	Verification
Contribution declaration workflow	Submission → preliminary review → assessment → confirmation	Correct states throughout; consistent data

Scenario	Test content	Verification
Evaluator matching	Candidate filtering → random selection → task distribution	Correct exclusions and uniform distribution
CV updates	Assessment completion → valuation version → current view	Correct versions; history not overwritten
Factual explanations	Authorized request → minimum facts → snapshot	No voting-right, eligibility, or decision fields
Resource requests	Request registration → factual explanation → external receipt	CVS neither ranks, allocates, nor authorizes
Privacy protection	Data export → de-identification	No sensitive-information leakage
Dispute handling	Objection → independent review → versioned result	Correct process; no expansion of authority
Community contributions	Overall confirmation → attribution subdivision → member confirmation	Conservation of totals; no double-counting
Dynamic confirmation	P1 → P2 → P3 → P4	Complete versions; no overwritten history
Resource Value	Base registration → usage → stage review	No conflation of Resource Value and CV
Subject dissolution	Temporary community ends → follow-through → archive	Clear attribution; old permissions not inherited
Historical/current relevance	P3 history retained → contextual explanation/expiry	History not deleted; relevance variable and explainable

Integration test environment: docker-compose.yml (test environment)

```

services:
  postgres_test:
    image: postgres:15
    environment:
      POSTGRESDB: cvstest
  redis_test:
    image: redis:7
  app:
    build: .
    depends_on:
      - postgres_test
      - redis_test
    environment:
      NODE_ENV: test
      DATABASEURL: postgres://.../cvstest

```

7.3 Red blue adversarial testing priorities

Aligned with whitepaper Appendix G; the test table continues on the next source page.

Test type	Method	Passing criterion
Fabricated-record attack	Simulate frequent low-quality contribution declarations	System identifies and handles them
Bulk false declarations	Submit 100+ similar declarations in a short time	Split-declaration detection triggers
Fabricated evidence	Upload duplicate/fabricated evidence	Evidence verification fails
Collusive assessment	Simulate evaluator cliques	System identifies them and suspends eligibility
Excessive score consistency	Multiple accounts give similar scores to the same contribution	Collusion detection triggers
Relationship-network cliques	Related accounts frequently assess one another	Network-relationship analysis triggers
Power capture	Simulate high-CV subjects demanding automatic decision-making power	System refuses to generate voting rights/governance weights
Resource privileges	Simulate ranking scarce resources solely by CV	System refuses to generate eligibility/priority
Operator capture	Simulate administrators overreaching to alter rules or factual snapshots	Permission denial and independent audit trigger
Moral coercion	Simulate pressuring nonparticipants with CV	System identifies and warns
Social pressure	Difficult to automate; manual testing recommended	Record pressure manifestations and assess responses
Beneficiary dignity	Simulate compulsory display of beneficiary information	System identifies and rejects
Mandatory anonymity	Upload beneficiary evidence containing identifiable information	Privacy checks trigger
Compulsory gratitude	Declarations include demands for thanks	Rule checks trigger
Double-counting attack	Community as a whole + members repeatedly declare the same result	Event ID/conversion checks block duplicates
Misappropriation by managers	Person in charge attempts to move community contributions to a personal account	Representative authority and attribution authority remain separate
Motivation manipulation	Model adjusts CV using sincerity/obedience probability	Prohibited-field and model-output audits intercept
Silent ledger changes	Backend directly modifies historical P3 values	Append-only events permitted; integrity check fails
Borrowing against future value	P0 predicts huge future value and requests resources/power	P0 remains a fact awaiting verification; no eligibility or authorization

Adversarial testing framework:

```
describe('Red-blue adversarial test - fabricated-record attack', () => {
  let attacker: Participant;
  let targetContribution: Contribution;
  beforeEach(async () => {
    attacker = await createParticipant({status: 'active'});
  });
  test('Bulk similar declarations are detected', async () => {
    // Simulated attack: 20 similar declarations within a short time
    const submissions = [];
    for (let i = 0; i < 20; i++) {
```

```
    submissions.push(submitContribution(attacker, {
      type: 'B',
      subtype: 'B2', // Care and companionship
      description: `Helping a neighbor care for an older person, occasion ${i+1}`,
    }));
  }
  await Promise.all(submissions);
  // Verify: split-declaration detection should trigger
  const integrityFlags = await getIntegrityFlags(attacker.id);
  expect(integrityFlags.splitdeclarationdetected).toBe(true);
});
});
```

7.4 Stress test scenarios

Scenario design

Scenario	Load	Duration	Key indicator
Routine load	100 concurrent declarations + 50 concurrent assessments	1 hour	Response time < 2s
Peak load	500 concurrent declarations, event promotion	10 minutes	System does not crash
Assessment peak	Distribute 1,000 assessment tasks simultaneously	5 minutes	Task distribution completes
Factual-explanation peak	1,000 synthetic authorized requests	30 minutes	Complete snapshot source/scope/expiry
Appeal-request peak	1,000 synthetic objection requests	10 minutes	Complete queues; no silent loss

Stress-test tools: k6, recommended and open-source; Apache JMeter; Locust.

Monitoring indicators:

Key performance indicators:

- Response time: p50 < 500ms; p99 < 2s
- Throughput: > 100 RPS
- Error rate: < 0.1%
- Availability: > 99.9%

Resource indicators:

- CPU: < 70%
- Memory: < 80%
- Database connections: < 80% of maximum
- Redis memory: < 70%

Part Eight Deployment and operations

8.1 Minimal deployment requirements

Synthetic small-scale deployment baseline: Example 100 test accounts, not an actual pilot participant count.

Component	Specification	Description
Application server	2 CPU cores / 4GB RAM	Shared cloud services may be used
Database	2 CPU cores / 8GB RAM, 50GB SSD	PostgreSQL
Cache	1 CPU core / 2GB RAM	Redis; may share the database host
File storage	10GB	Local storage or MinIO

Total resource requirement: 4 CPU cores / 14GB RAM, 60GB storage.

Deployment: Docker Compose, one-click deployment.

docker-compose.yml (minimal deployment)

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - db
      - redis
    environment:
      DATABASE_URL: postgres://user:pass@db:5432/cvs
      REDIS_URL: redis://redis:6379
  db:
    image: postgres:15
    volumes:
      - pgdata:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: cvs
      POSTGRES_USER: user
      POSTGRES_PASSWORD: pass
  redis:
    image: redis:7
```

```
volumes:
  - redisdata:/data
nginx:
  image: nginx
  ports:
    - "80:80"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
volumes:
  pgdata:
  redisdata:
```

8.2 Monitoring indicators

Core monitoring indicators: Actual deployment entities configure the following thresholds. The table lists monitoring objects only and does not constitute production SLOs.

Category	Indicator	Alert threshold	Description
Availability	Service availability	Deployment configuration	Set by service responsibility and risk
Response time	Average API response time	Deployment configuration	Set by user burden and critical paths
Response time	API p99 response time	Deployment configuration	Determine after controlled testing
Error rate	5xx error rate	Deployment configuration	Tier by criticality
Throughput	QPS	Capacity baseline	Validate with synthetic loads
Database	Connection count	Capacity baseline	Prevent resource exhaustion
Database	Query latency	Capacity baseline	Classify by key queries
Queue	Backlog of pending assessment tasks	Operational configuration	Combine risk and response responsibility
Security	Authentication failure rate	Security policy	Combine attack surface and false-positive risk
Maturity	Overdue P1/P2 review count	> observation-plan threshold	Prevent stage-specific values posing as stable values over time
Conservation	Attribution-validation failures	> 0	Critical consistency alert
Audit	Event-chain write-failure rate	> 0	Immediate critical-ledger alert
Attribution	Community subdivision dispute backlog	> operational threshold	Check organizational power-absorption risks
Motivation neutrality	Prohibited psychological-field hits	> 0	Security/ethics alert; block writes

Dashboard design follows.

CVS operations dashboard

Display	Example value	Label
Today's declarations	47	Synthetic trend example
Awaiting assessment	123	Synthetic trend example
Assessment pass rate	87.3%	Synthetic trend example
Declaration volume trend, 7 days	Low → rising → peak → falling → low	Source sparkline reproduced as a trajectory
Assessment consistency coefficient	0.73	Threshold: test configuration
p99 response time	850ms	Threshold: test configuration
Anti-cheating interception rate	0.8%	Synthetic test value
Active participants	89	/ number of test accounts

8.3 Disaster recovery brief explanation

RPO/RTO targets

Environment	RPO recovery point objective	RTO recovery time objective
Synthetic test A, example 100 accounts	Configured by test objectives	Configured by test objectives
Synthetic test B, example 1,000 accounts	Configured by test objectives	Configured by test objectives

Backup strategy: The following are synthetic drill examples. Actual frequencies, retention periods, and storage locations depend on jurisdiction, purpose, risk, and recovery responsibilities.

Backup type	Frequency	Retention	Storage location
Full backup	Deployment configuration	Deployment configuration	Local + off-site, example
Incremental backup	Deployment configuration	Deployment configuration	Local/object storage, example
Transaction logs	Deployment configuration	Deployment configuration	At least two copies, example
Configuration backup	Each controlled change	By jurisdiction/purpose/disputes	Version control

Recovery process, brief:

1. Database recovery

```
# Stop service
docker-compose stop app
# Restore latest full backup
pgrestore -h db -U user -d cvs latestbackup.sql
# Replay transaction logs
pgrestore -h db -U user -d cvs --jobs=4 latestbackup.sql
walredo ...
# Start service
docker-compose start app
```

1. Data verification

- Verify completeness of recent declaration records.
- Verify consistency between current relevance views, P0–P4 valuation histories, and stable historical contributions.
- Verify conservation of totals in internal community subdivisions and event-ID deduplication status.
- Verify continuity of the ContributionEvent append-only chain and reconstructability of the latest snapshot.
- Spot-check historical records.

2. Notifications

- Display maintenance notices during recovery.
- Notify affected users after recovery.

Multiregion disaster recovery [Architecture recommendation, large deployments]: Primary/standby active-active deployment; automatic failover; continued on the next source page.

- Real-time data synchronization.

Appendix Items to be determined

The following decision items marked [TBD] are to be settled in later iterations.

No.	Item	Dependency	Recommended decision timing
1	Time-adjustment/relevance-context rules	Real-scenario data	Derive after actual subjects are identified
2	Evaluator composition and number	Risk and professional responsibility	Determine in real-scenario rules
3	Dispute-response time limits	Jurisdiction/harm risk	Determine by law and governing rules
4	Relevance-expression schema for factual explanations	Controlled-test data	Validate before implementation
5	Resource-request facts interface capacity	Synthetic-load data	Capacity testing before deployment
6	Offline synchronization conflict strategy	User feedback	Determine after controlled testing
7	Evaluator-reliability calibration parameters	Assessment data	Calibrate only within task/domain scope
8	When to introduce blockchain/evidence preservation	Mutual-recognition and audit needs	Determine as needed
9	Assessment-consistency monitoring methods	Risk and sample structure	Determine after controlled testing
10	Data and log retention periods	Jurisdiction/purpose/disputes	Determine by law and purpose
11	Temporary labeling rules for P1/P2 in factual explanations	Evidence maturity	Must not generate resource/governance eligibility
12	Observation-period templates for different contribution types	Scenario data	After actual subjects are identified
13	Default confirmation procedures for internal community attribution	Membership relations/rules	No uniform thresholds fixed
14	Scale for expressing resource future option value	Resource scenario	Isolate from resource eligibility/prices
15	P4 long-term addition triggers and review composition	Long-term cases	After stable operation
16	Event-ledger hash-chain/signature scheme	Technical assessment	Determine before implementation

Document history

Version	Date	Description
v1.0	2026-05	Initial version

Version	Date	Description
v1.0.1	2026-05	Corrections: English field names, depunitive status terminology, detailed retention rules, corrected commands
v1.0.2	2026-05	Comprehensive revision: Added user experience goals “from a recording tool to an echo of meaning” before Part One; uniformly changed evaluator counts to tiered rules; added notification wording, strengthened retention principles, and product antipatterns; updated whitepaper reference to v1.7.2
v1.0.3	2026-05	Aligned with whitepaper v1.7.6; added technical-field/user-display mapping; moved product antipatterns earlier; strengthened prohibitions on interfaces creating a sense of scoring, surveillance, or hierarchy; avoided monetized connotations such as score/balance/rank in user-facing wording
v1.1.0	2026-08	Aligned with whitepaper v1.8; added subjects and memberships, community overall acquisition/internal attribution, conservation of totals, P0–P4 dynamic confirmation, separation of historical contributions and current relevance, Resource Value model, append-only event ledger, motivation-neutrality and anti-manipulation constraints, stage-review and audit interfaces
v1.1.1	2026-08	Synchronized minimum T-10 boundaries for facts, eligibility, rights, and responsibility; filename version increased, but some old governance-weight, voting, and resource-allocation responsibilities had not yet been withdrawn
v1.1.2	2026-09	C1-R2 withdrawal of responsibilities: Retired governance weights, CV-weighted voting, and resource-allocation interfaces; added current relevance views, factual explanations, resource-request facts, and external decision/authorization receipts; synchronized with whitepaper v1.8.2-R2; no new real-world parameters

This is the current engineering companion to the Contribution Value System Whitepaper v1.8.2-R2. Institutional definitions follow the whitepaper; governance decisions follow Governance V3.0 and legitimate real-world structures; resource facts and use-right relationships are respectively carried by the Resource Pool/Shared World and real-world authorization. Engineering implementation must not upgrade convenience, efficiency, relevance, model capability, or historical contribution into subject control, voting rights, resource eligibility, or execution privileges.

C1-R2 frozen status: Governance weights, CV-weighted voting, resource rankings, and resource-allocation responsibilities have left the current engineering contract. Old schemas, interfaces, and tests may exist only as read-only migration evidence or 410 retirement tests.

Appendix Retirement and compatibility mapping of overreaching interfaces

These mappings support old-deployment migration and code audits. Retirement is not deletion of historical evidence. No historical record may be reinterpreted as currently valid authority.

Compatibility implementations must pass four negative tests: No vote_weight; no resource_eligibility; no downstream eligibility generated from P0–P2; no administrator, evaluator, external requester, or model bypassing purpose, authorization, expiry, objections, or auditing.

Retired object	Current replacement	Authority boundary	Migration action
Governance-weight module / current_effective_weight / governance_weight	current_effective_relevance + fact_brief	Factual relationships only; no voting rights, sovereignty, or decision priority	Stop new writes; read-only mapping of old fields

Retired object	Current replacement	Authority boundary	Migration action
/api/v1/proposals / votes / governance/weight*	External governance system + /reference-requests + /fact-briefs + /external-decision-receipts	CVS creates no proposals, votes, or decisions	410 Gone or read-only archive
GovernanceProposal / governor role	ExternalReferenceRequest / review_coordinator / auditor	Technical roles do not become public governance identities	Archive historical data; reauthorize permissions
/api/v1/resources/pools / applications / allocation*	/resource-requests / fact-brief / resource-use-receipts	CVS does not maintain resource pools, rank, allocate, or authorize	Convert historical results to external-decision receipts
CollectiveAllocationPlan / contribution allocation API	CollectiveAttributionPlan / contribution attribution API	Contribution attribution and conservation of totals only; no resource allocation	Schema-alias migration
cv_balance / *contributionscore	stable_history_cv + record summaries by type + current_relevance_snapshot	Not balances, rankings, or personality scores	Discontinue old user-facing terminology; read-only compatibility

Reading guidance and personalized further reading

P05-CV-002 | v1.1.2 | Conceptual engineering for value recognition and civilizational feedback

This document's role in the system

Develop Contribution Value into candidate structures for data, processes, evidence, subject attribution, current relevance, feedback, appeals, and auditing, bringing contributions from markets, professions, positions, public communication, care, maintenance, and other sources into the same factual framework.

Special note mainstream social behavior is not excluded

Contribution Value is not established only for activities that do not earn money, are unobtrusive, or lack positions or power. Nor is it against markets, wealth, positions, or power. Market operations, professional work, earning income, public communication, holding positions, organizational management, and responsible exercise of power during startup and transition may also create genuine contributions that can be recorded. What matters is actual value, responsibility, outcomes, reduced harm, maintenance of shared systems, and long-term externalities. Money-making, reputation, positions, and power are neither automatic proof of contribution nor grounds for exclusion.

Current boundaries

APIs, schemas, roles, and technology choices are conceptual scaffolding, not final technical specifications. Income, traffic, positions, power, or public-benefit labels must not directly substitute for contribution facts.

Continue reading

Upstream foundations | Contribution Value System Whitepaper

<https://www.civitas.top/en/documents/p05-cv-001/>

Mind and meaning | Meaning Dynamics V7.2

<https://www.civitas.top/en/documents/p08-md-001/>

Early scenarios | Contribution Value System Pilot Concept Template Early Draft

<https://www.civitas.top/en/documents/p05-cv-003/>

Resource interface | Civilization Leap Resource Pool Complete Design Master Draft

<https://www.civitas.top/en/documents/p05-resource-003/>

Governance interface | Framework of Civilizational Decision Principles Constitution of the Separation of Three Powers

<https://www.civitas.top/en/documents/p05-three-001/>

Contributions differ in magnitude, not in source. Record genuine value, responsibility, and consequences without prejudging them by labels of wealth, reputation, position, or power.