CIV-CARE CIV-CALL

DE KAMER VAN SOCIALE WAARDEN

Menselijke waardigheid als fundament

CIV-CAMP CIV-RAMP

Praktische Uitwerking 5

IT-Architectuur: Technische specificaties RC-tracking systeem Geconsolideerde heruitgave v0.901

(SSI-native, originele hoofdstukindeling behouden)

Datum: 15 February 2026
Regeneratief Economisch Model
Stichting De Kamer van Sociale Waarden

Opgesteld door:

Alexander Groenheide
Voorzitter
Stichting De kamer van Sociale Waarden
Naar de waardevolle Samenleving
info@dekvsw.nl
www.dekvsw.nl
06 53 44 50 54
Laurastraat 87, 6471 JJ Eygelshoven
KVK: 97098817 | RSIN: 867909274
W: DeKvSW.nl | ubuntukids.nl |
civ-care.nl | civ-call.com | civ-ramp.nl | civ-camp.nl

Samen bouwen aan een waardegedreven samenleving
Liefde – Samenwerking – Zorg voor elkaar

Inhoudsopgave

Inhoud

1. Doel en Reikwijdte

Dit document specificeert de technische architectuur van het RC-tracking systeem. In versie 0.901 is Self-Sovereign Identity (SSI) het primaire identiteitsmodel. Federatieve authenticatiesystemen (zoals DigiD) kunnen uitsluitend als optionele verificatiepoort worden toegepast.

2. Architectuurprincipes

- SSI-first: geen centrale identity provider of PII-database.
- Dataminimalisatie en privacy-by-design (AVG art. 25).
- Cryptografisch verifieerbare logs (auditability-by-design).
- Selective disclosure en proportionaliteit.
- Falsificeerbaarheid via meetpunten en stopcriteria.

3. Systeemarchitectuur – Overzicht

Het systeem bestaat uit vier functionele lagen:
1) Identiteit & Credentials (SSI: DID/VC/wallet)
2) Attestatie & Validatie (proof verification + policy engine)
3) RC Ledger (pseudonieme event-sourced transacties)
4) Governance & Audit (audit node, dashboards, incidentrespons)
Componenten:
- SSI Wallet (beheer DID, sleutels, VC's).
- Credential Issuers (gemeente, instelling, examencommissie).
- Verifier/Policy Engine.
- RC Ledger Service.
- Audit Node.
- Public Dashboard (geaggregeerd, geen PII).

4. RC-Ledger Specificaties

Het ledger is append-only en event-sourced. Er worden geen persoonsgegevens opgeslagen. Transacties verwijzen uitsluitend naar DID's en gehashte payloads.
Minimum event types:
- claim_submitted
- claim_validated
- attestation_added
- allocation_decision_recorded
- rc_awarded
- rc_revoked (indien toegestaan)
- credential_status_checked
- incident_flagged

5. Validatie- en Auditmechanismen

Validatie vindt plaats via verificatie van Verifiable Presentations (VP's) inclusief controle op credential status (revocation/status lists).
Audit logs zijn hash-chained en periodiek ge-anchored. Steekproefsgewijze audits (5–20%) worden uitgevoerd door een onafhankelijke audit node.

6. Gegevensbescherming en Compliance

Het systeem slaat geen PII op. VC's worden niet volledig bewaard; alleen hashes en statusreferenties. Selective disclosure minimaliseert gegevensdeling.

7. Beveiligingsmaatregelen

Authenticatie geschiedt via signed requests en DID-verificatie; geen session-based login endpoints.
Sleutelbeheer ondersteunt key rotation en recovery (social recovery, multi-sig).

8. Governance en Toezicht

Gemeenten en bevoegde instanties kunnen optreden als credential issuers, maar beheren geen centrale identiteitsdatabase.
Besluiten worden vastgelegd met motiveringshashes; toezicht vindt plaats via audit nodes en transparante dashboards.

9. Schaalbaarheid en Performance

Beschikbaarheid: 99,5% (pilot), 99,9% (opschaling).
Event-sourcing maakt horizontale schaalbaarheid mogelijk.
VC-verificatie wordt geoptimaliseerd via caching van DID-documents.

10. Implementatiefasen

Fase 1 – SSI Pilot
Fase 2 – Evaluatie sleutelbeheer en UX
Fase 3 – Gefaseerde uitrol
Fase 4 – Externe audit en optimalisatie

11. Risicoanalyse

Stopcriteria:
- Sleutelverlies >15% zonder herstel (rolling 90d).
- Fraude false-negative >2% (audit sample).
- Interoperabiliteitsfouten >1%.
- Extra administratieve last >5 min per claim.
- Onboarding median >20 minuten.

12. Conclusie

Versie 0.901 consolideert de identiteitlaag conform SSI-principes, met behoud van de oorspronkelijke hoofdstukindeling van v0.9. Het systeem is technisch sluitend, privacy-by-design en auditbaar.

BIJLAGE A – UML-Architectuurschema's (PlantUML) – SSI-native

A1. Componentdiagram

```
@startuml
title A1. Componentdiagram (SSI-native RC-tracking)

package "Client" {
  [SSI Wallet] as Wallet
}

package "Issuers" {
  [Gemeente VC Issuer] as IssuerMunicipality
  [Institutionele VC Issuer] as IssuerOrg
  [Examencommissie VC Issuer] as IssuerExam
  [Revocation/Status List] as StatusList
}

package "Core Services" {
  [Verifier + Policy Engine] as Verifier
  [Attestation Service] as Attest
  [RC Ledger Service] as Ledger
  [Audit Log Service] as AuditLog
}

package "Oversight" {
  [Audit Node] as AuditNode
  [Public Dashboard (Aggregated)] as Dashboard
}

Wallet --> Verifier : VP (Proofs)
Verifier --> StatusList : status check
Verifier --> Attest : record attestation
Verifier --> Ledger : append RC events
Ledger --> AuditLog : hash-chained log
AuditNode --> Ledger : verify integrity
AuditNode --> AuditLog : verify hashes
Dashboard --> Ledger : aggregated queries

IssuerMunicipality --> Wallet : VC.Residency / VC.Role
IssuerOrg --> Wallet : VC.Professional
IssuerExam --> Wallet : VC.TrainingCertificate
@enduml
```

A2. Sequencediagram (claim tot toekenning)

```
@startuml
title A2. Sequencediagram (claim tot toekenning) – SSI-native

actor "Burger/Zorgprofessional" as User
participant "SSI Wallet" as Wallet
participant "Verifier+Policy" as Verifier
participant "Attestation Service" as Attest
participant "Wijkraad Besluitvorming" as Council
participant "RC Ledger" as Ledger
participant "Audit Log" as Audit

User -> Wallet : Start claim
Wallet -> Verifier : Submit VP + claim payload (signed)
Verifier -> Verifier : Verify VP (DID/VC signatures)
Verifier -> Verifier : Check revocation/status
Verifier -> Attest : Request/record peer attestations
Attest --> Verifier : Attestations confirmed (hash refs)
Verifier -> Council : Prepare decision dossier (hashes, criteria)
Council --> Verifier : Decision (approve/deny) + motivation hash
Verifier -> Ledger : Append events (validated, decision, rc_awarded)
Ledger -> Audit : Append hash-chained audit entry
Verifier --> Wallet : Result + tx reference (no PII)
@enduml
```

A3. Klassendiagram (kern-entiteiten)

```
@startuml
title A3. Klassendiagram (kern-entiteiten) – SSI-native

class DID {
  +did: string
  +createdAt: datetime
  +status: enum
}

class VerifiableCredential {
  +id: string
  +type: string
  +issuer: string
  +issuanceDate: datetime
  +expirationDate: datetime
  +credentialSubjectHash: string
  +statusRef: string
}

class VerifiablePresentation {
  +id: string
  +holderDid: string
  +proof: string
  +presentedAt: datetime
}

class RCClaim {
  +claimId: string
  +subjectDid: string
  +category: enum
  +payloadHash: string
  +submittedAt: datetime
}

class Attestation {
  +attestationId: string
  +claimId: string
  +attestorDid: string
  +attestationHash: string
  +createdAt: datetime
}

class AllocationDecision {
  +decisionId: string
  +claimId: string
  +councilDid: string
  +result: enum
  +motivationHash: string
  +decidedAt: datetime
}

class RCTransactionEvent {
  +eventId: string
  +type: enum
  +subjectDid: string
  +counterpartyDid: string
  +amount: int
  +eventHash: string
  +timestamp: datetime
}

class AuditEntry {
  +entryId: string
  +eventId: string
  +prevHash: string
  +hash: string
  +timestamp: datetime
}

DID "1" -- "many" VerifiableCredential : holder
```

```
VerifiablePresentation "1" -- "many" VerifiableCredential : includes
RCClaim "1" -- "many" Attestation
RCClaim "1" -- "1" AllocationDecision
RCClaim "1" -- "many" RCTransactionEvent
RCTransactionEvent "1" -- "1" AuditEntry
@enduml
```

BIJLAGE B – Datamodel (ERD) en Tabellen – SSI-native

B1. ERD – Tabellen en relaties (tekstueel)

```
ERD (event-sourced + minimal relational index)

TABLE did_registry (
  did                 VARCHAR PK,
  status              VARCHAR NOT NULL,       -- active/suspended/rotated
  created_at          TIMESTAMP NOT NULL,
  updated_at          TIMESTAMP NOT NULL
)

TABLE vc_hash_store (
  vc_id               VARCHAR PK,
  holder_did          VARCHAR NOT NULL FK -> did_registry.did,
  issuer_did          VARCHAR NOT NULL,
  vc_type             VARCHAR NOT NULL,
  vc_hash             CHAR(64) NOT NULL,      -- hash of VC payload
  status_ref          VARCHAR NOT NULL,       -- status list URL/ref
  issued_at           TIMESTAMP NOT NULL,
  expires_at          TIMESTAMP NULL
)

TABLE rc_claims (
  claim_id            VARCHAR PK,
  subject_did         VARCHAR NOT NULL FK -> did_registry.did,
  category            VARCHAR NOT NULL,
  payload_hash        CHAR(64) NOT NULL,
  submitted_at        TIMESTAMP NOT NULL,
  state               VARCHAR NOT NULL        -- submitted/validated/decided/awarded/denied
)

TABLE attestations (
  attestation_id      VARCHAR PK,
  claim_id            VARCHAR NOT NULL FK -> rc_claims.claim_id,
  attestor_did        VARCHAR NOT NULL FK -> did_registry.did,
  attestation_hash    CHAR(64) NOT NULL,
  created_at          TIMESTAMP NOT NULL
)

TABLE allocation_decisions (
  decision_id         VARCHAR PK,
  claim_id            VARCHAR UNIQUE NOT NULL FK -> rc_claims.claim_id,
  council_did         VARCHAR NOT NULL FK -> did_registry.did,
  result              VARCHAR NOT NULL,       -- approve/deny
  motivation_hash     CHAR(64) NOT NULL,
  decided_at          TIMESTAMP NOT NULL
)

TABLE rc_events (
  event_id            VARCHAR PK,
  claim_id            VARCHAR NULL FK -> rc_claims.claim_id,
  event_type          VARCHAR NOT NULL,       -- claim_validated/rc_awarded/...
  subject_did         VARCHAR NOT NULL FK -> did_registry.did,
  counterparty_did    VARCHAR NULL FK -> did_registry.did,
  amount              INT NULL,
  event_hash          CHAR(64) NOT NULL,
  created_at          TIMESTAMP NOT NULL
)

TABLE audit_log (
  entry_id            VARCHAR PK,
  event_id            VARCHAR NOT NULL FK -> rc_events.event_id,
  prev_hash           CHAR(64) NOT NULL,
  hash                CHAR(64) NOT NULL,
  created_at          TIMESTAMP NOT NULL
)
```

```
NOTES:
- Geen PII-tabellen (geen users/email/BSN).
- VC's worden niet volledig opgeslagen; alleen hashes + statusrefs.
- Presentations/proofs worden niet bewaard, enkel verificatieresultaten/audit entries indien
nodig.
```

B2. Constraints en datakwaliteit

```
CONSTRAINTS (selectie):
1) rc_claims.payload_hash UNIQUE per (subject_did, category, time_window)  -> dubbele claims
detectie
2) attestations: (claim_id, attestor_did) UNIQUE                           -> één attestatie
per attestor
3) allocation_decisions.claim_id UNIQUE                                    -> één besluit per
claim
4) rc_events: event_hash UNIQUE                                             ->
immutability/integriteit
5) audit_log: hash = SHA256(prev_hash + event_id + event_hash + created_at)-> hash chain
6) did_registry.status transitions gecontroleerd (active->rotated, active->suspended)
7) VC status check verplicht vóór award (audit event 'credential_status_checked' of policy
flag)

DATA QUALITY RULES:
- Tijdstempels in UTC, monotone ordering per claim.
- Policy Engine logt reason codes bij afwijzing/flagging.
- Dashboard gebruikt alleen geaggregeerde query's met k-anonimiteit (k>=10).
```

BIJLAGE C – Formele API-specificaties (OpenAPI 3.0) – SSI-native

C1. OpenAPI 3.0 YAML (kern-endpoints)

```
openapi: 3.0.3
info:
  title: RC Tracking API (SSI-native)
  version: "0.901"
servers:
  - url: https://api.rc.example/v1
security:
  - SignedRequest: []
paths:
  /rc/claims:
    post:
      summary: Submit RC claim with Verifiable Presentation (VP)
      security:
        - SignedRequest: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ClaimSubmission'
      responses:
        "202":
          description: Accepted
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ClaimReceipt'
  /rc/claims/{claimId}:
    get:
      summary: Get claim status (no PII)
      parameters:
        - in: path
          name: claimId
          required: true
          schema: { type: string }
      responses:
        "200":
          description: Claim status
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ClaimStatus'
  /rc/attestations:
    post:
      summary: Submit attestation for a claim (signed)
      security:
        - SignedRequest: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AttestationSubmission'
      responses:
        "202":
          description: Accepted
  /rc/ledger/events:
    get:
      summary: Query ledger events (aggregated or per DID with proof)
      parameters:
        - in: query
          name: subjectDid
          schema: { type: string }
        - in: query
          name: from
          schema: { type: string, format: date-time }
        - in: query
          name: to
          schema: { type: string, format: date-time }
      responses:
        "200":
          description: Events
          content:
            application/json:
```

```
              schema:
                $ref: '#/components/schemas/LedgerEventList'
  /credentials/status/check:
    post:
      summary: Check credential status (revocation/status list)
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CredentialStatusCheck'
      responses:
        "200":
          description: Status result
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/CredentialStatusResult'
components:
  securitySchemes:
    SignedRequest:
      type: apiKey
      in: header
      name: X-Signature
      description: >
        Detached signature over request body using holder DID key.
        Header set includes X-DID and X-Signature; server verifies via DID resolution.
  schemas:
    ClaimSubmission:
      type: object
      required: [vp, claim]
      properties:
        vp:
          type: object
          description: Verifiable Presentation (W3C) with selective disclosure proofs
        claim:
          type: object
          required: [category, payloadHash, subjectDid]
          properties:
            subjectDid: { type: string }
            category: { type: string }
            payloadHash: { type: string, description: SHA-256 hash of claim payload }
    ClaimReceipt:
      type: object
      properties:
        claimId: { type: string }
        acceptedAt: { type: string, format: date-time }
        status: { type: string }
    ClaimStatus:
      type: object
      properties:
        claimId: { type: string }
        state: { type: string }
        reasonCodes: { type: array, items: { type: string } }
        lastUpdated: { type: string, format: date-time }
        txRef: { type: string, nullable: true }
    AttestationSubmission:
      type: object
      required: [claimId, attestorDid, attestationHash]
      properties:
        claimId: { type: string }
        attestorDid: { type: string }
        attestationHash: { type: string }
    LedgerEventList:
      type: object
      properties:
        events:
          type: array
          items:
            $ref: '#/components/schemas/LedgerEvent'
    LedgerEvent:
      type: object
      properties:
        eventId: { type: string }
        eventType: { type: string }
        subjectDid: { type: string }
        counterpartyDid: { type: string, nullable: true }
        amount: { type: integer, nullable: true }
        createdAt: { type: string, format: date-time }
        eventHash: { type: string }
    CredentialStatusCheck:
      type: object
      required: [vcId, statusRef]
```

```
      properties:
        vcId: { type: string }
        statusRef: { type: string }
    CredentialStatusResult:
      type: object
      properties:
        vcId: { type: string }
        status: { type: string, description: valid/revoked/expired/unknown }
```

C2. API-regels (security, audit, versioning)

```
SECURITY RULES:
- Geen /login, /register endpoints. Authenticatie = signed request + DID verification.
- Verifiers controleren VP proofs + VC status (revocation/status list) vóór toekenning.
- Rate limiting per DID (sliding window), plus anomaly detection (burst claims).
- Replay protection: nonce + timestamp in signed headers; server enforces freshness.

AUDIT RULES:
- Elke state transition (submitted->validated->decided->awarded) logt reason codes + hashes.
- Audit log is hash-chained; dag-anchoring in extern register (optioneel) voor tamper
evidence.
- Sample-based audit: 5-20% claims (policy-config) met diepere verificatie.

VERSIONING RULES:
- Semver: v0.901 (patch) => SSI-consistentie, geen functionele scope uitbreiding buiten
identity/data/api.
- API versioning via /v1 en OpenAPI version field.
- Backward compatibility: oude auth endpoints blijven NIET bestaan; migratie via wallet
onboarding en issuer issuance.

PRIVACY RULES:
- PII verboden in request payloads en database (contractueel + technische validators).
- Dashboards tonen alleen geaggregeerde statistiek met k-anonimiteit.
```