This document provides an approach to codify regulations

To reliably certify processes and systems against real‐world laws and standards, we need to turn each regulation—its clauses, definitions, obligations, and penalties—into precise, machine-readable rules that your toolkit can execute. Below is a step-by-step prescription for **codifying** regulations into your Compliance Certification Toolkit:

**1. Regulation Decomposition**

1. **Identify Scope and Structure**
   * Break the regulation document into its top-level sections (e.g., “General Principles,” “Data Subject Rights,” “Enforcement”).
   * For each section, list its articles/clauses, definitions, and any sub-clauses.
2. **Extract Key Elements**
   * **Definitions**: Extract and normalize terms (“Personal Data,” “Data Controller,” etc.) into a shared vocabulary.
   * **Obligations**: Statements prescribing required actions (“must,” “shall,” “will”).
   * **Prohibitions**: Statements forbidding actions (“shall not,” “must not”).
   * **Conditions**: Preconditions or triggers (“if,” “where,” “when”).
   * **Remedies & Penalties**: Note any mandated responses for non-compliance.

**2. Define a Policy DSL (Scroll Schema)**

Design a small, declarative schema to capture each clause. For example, in YAML:

yaml

CopyEdit

id: GDPR-5.1

title: "Data Minimization"

description: >

Personal data shall be adequate, relevant and limited to what is necessary

in relation to the purposes for which they are processed.

definition:

- "Personal Data": "Any information relating to an identified or identifiable person"

conditions:

- context.data\_fields.max() <= context.required\_fields.max()

obligation: "Enforce collection limits at data ingestion"

weight: 4

remediation: "Drop any extra fields beyond the minimal set"

Fields to include in your scroll schema:

* **id**: Unique clause identifier
* **title**: Short human-readable name
* **description**: Full text of the clause
* **definitions**: Term mappings for context
* **conditions**: Logical expressions over your context model
* **obligation**: Action your engine must check or enforce
* **weight**: Relative importance for scoring
* **remediation**: Suggested fix or escalation path

**3. Build a Context Model**

Your engine needs a standardized “world state” to evaluate conditions. Define:

| **Context Variable** | **Type** | **Source** |
| --- | --- | --- |
| data\_fields | List[str] | Schema registry or process metadata |
| required\_fields | List[str] | Business process definition |
| consent\_recorded | bool | Audit logs or consent database flag |
| retention\_period\_days | int | Data policy configuration |
| … | … | … |

This model bridges real-world artifacts (databases, logs) to your scroll conditions.

**4. Translate Clauses into Scrolls**

1. **Author YAML Scrolls**
   * For each clause, write a YAML file following your DSL.
   * Group scrolls by regulation (folder per regulation).
2. **Version & Review**
   * Store scrolls in Git with semantic versioning (e.g., gdpr/v1.2/5.1-data-minimization.yaml).
   * Engage legal/compliance SMEs to validate logic and weights.
3. **Automated Validation**
   * Write unit tests that feed controlled context fixtures into your ScrollRunner and assert the expected pass/fail.

**5. Define Simulation Forks**

For decision points that depend on system or process behavior (e.g., “Has consent been refreshed yearly?”):

yaml

CopyEdit

- description: "Consent up to date"

condition: "context.last\_consent\_date > context.today - timedelta(days=365)"

remediation: "Trigger user re-consent workflow"

weight: 3

Your ForkEvaluator consumes these to chart out “compliant” vs “non-compliant” branches and captures the decision trail.

**6. Mirror & Scoring Rules**

* **Scroll Score**: Sum of passed scroll weight ÷ total weights.
* **Fork Score**: Percentage of forks evaluated to True.
* **Composite Score**: Weighted average, e.g.

ini

CopyEdit

overall\_score = 0.7 \* scroll\_score + 0.3 \* fork\_score

Document these formulas in your spec, so auditors understand how numerical ratings are derived.

**7. Reporting & Audit Trail**

* **Trace Logs**: For every evaluation, record
  + Scroll ID → pass/fail → timestamp
  + Fork ID → outcome → remediation suggestion
* **Certification Report**: Display clause-level results, overall score, next steps.

Keep all logs immutable (e.g., append-only JSON or blockchain-backed) for post-audit requirements.

**8. Continuous Updates**

* **Regulation Watch**: Monitor official sources for amendments.
* **Scroll Update Pipeline**:
  1. SME authors updated YAML scroll.
  2. CI runs unit tests against sample contexts.
  3. Publish new scroll versions and trigger re-certifications if needed.

**🛠️ Sample Workflow**

1. **SME** extracts Article 5 clauses → writes gdpr/5.1-data-minimization.yaml.
2. **Engineer** maps data model fields → ensures context.data\_fields.
3. **CI** validates scroll logic against test contexts.
4. **Toolkit** loads updated scrolls → re-runs certification for affected systems → issues updated certificates.

By structuring regulations into a clear YAML DSL, mapping to a shared context, and building simulation forks + scoring rules, we are fully codifying “laws and rules” into an executable compliance toolkit—providing transparency, auditability, and automated certification.