From Regulatory Text to Running Code

Policy-as-Code for Federal Environmental Compliance

The Challenge

Federal agencies face persistent dilemmas when implementing regulatory programs. Policy manuals run to hundreds of pages. Business rules are embedded in dense regulatory prose. Technical teams must translate requirements like “the proposed action shall not significantly affect unique characteristics of the geographic area” into functional software.

The challenges compound across the implementation chain. Applicants must assemble input data and supporting documentation without always understanding which regulatory provisions apply. Agency decision makers must then interpret complex regulations consistently—a particularly difficult task when determinations must align across dozens of state offices and multiple program areas operating under different timelines and priorities.

This translation process is costly and slow. Policy updates require new development cycles. Interpretation questions demand manual review of regulation text. When new versions of regulations are published, implementation often begins from scratch. The result: weeks-long determination processes, variable outcomes for similar projects, and mounting backlogs when regulations change.

There is a more efficient approach.

CADMUS

The Core Insight

Federal regulations already contain structured logic, it simply needs to be made explicit. Consider this regulatory text from Natural Resource Conservation Service (NRCS) environmental compliance:

> "Replacing and repairing existing culverts, grade stabilization, and water control structures and other small structures that were damaged by natural disasters where there is no new depth required and only minimal dredging, excavation, or placement of fill is required."
> — ***7 CFR § 1b.4(d)(6)***

This single sentence encodes conditional logic:

IF the structures are existing (not new construction) **AND**

IF damage resulted from natural disasters **AND**

IF no new depth is required **AND**

IF only minimal dredging/excavation/fill is needed **THEN**

The action may qualify for categorical exclusion under this provision.

Our approach makes this implicit logic explicit and machine-executable. Rather than translating regulations into software after the fact, we structure policy rules in formats that both humans and computers can interpret directly.

The Breakthrough: Converting Human-Readable Policy to Machine-Executable Rules

We solved this problem years before artificial intelligence (AI) language models existed.

The Centers for Medicare & Medicaid Services (CMS) publishes patient assessment specifications for Long-Term Care Hospital (LTCH), Outcome and Assessment Information Set (OASIS), and Inpatient Rehabilitation Facility (IRF) programs. These specifications determine quality measures, public reporting, and payment calculations. They change annually. They are written in English prose, not code.

The traditional approach embedded validation logic directly in software. Each annual specification update required months of development: several developers, multiple months of coding, extensive regression testing, and delayed availability to healthcare providers. When specifications changed, developers rewrote thousands of lines of validation code. Supporting multiple versions simultaneously meant maintaining complex conditional logic across different codebases.

We built a different solution: the **Spec Instrumentation Framework (SIF)**.

The core innovation was the **SIF Lexer℠**, a specification parsing engine that reads human-readable English text and transforms it into machine-processable JSON validation schemas. through deterministic linguistic pattern recognition and rule extraction.

How the Lexer Works: Pattern Recognition in Regulatory Language

The Lexer recognizes that regulatory specifications, despite appearing as unstructured prose, follow consistent linguistic patterns:

Pattern: "Field X must contain a value between 0 and 100"
Lexer extracts: Numeric range constraint ➡ { "min": 0, "max": 100 }

Pattern: "This field is required when patient age is 65 or older"
Lexer extracts: Conditional requirement ➡ { "if": "age >= 65", "then": "required" }

> "Rather than translating regulations into software after the fact, we structure policy rules in formats that both humans and computers can interpret directly.

Pattern: “Valid values are: 01-Acute, 02-Chronic, 03-Both”
Lexer extracts: Enumerated list ➡ { “valid_values”: [“01”, “02”, “03”] }

Pattern: “Item B0100 must be completed unless C1610 equals 0”
Lexer extracts: Cross-field dependency ➡ { “required_unless”: “C1610 == 0” }

The Lexer identifies signal words and phrases—“must,” “shall,” “required when,” “unless,” “between,” “valid values are”—and maps them to schema structures. Date validations, cardinality rules, and conditional logic are all extracted programmatically from regulatory prose.

The result: validation logic lives outside code. Specifications are maintained as structured CSV files. The SIF Lexer℠ transforms them into JSON schemas. A validation engine processes assessment files against these schemas with no code changes required. When CMS publishes updated specifications, we run the Lexer, generate new schemas, and deploy. The entire update cycle is days, not months.

> “What the SIF Lexer℠ does through pattern-matching rules, AI does through context-aware understanding.

Multiple specification versions (V1, V2, V3) run concurrently. Version 2.0 assessments validate against v2.0 schemas. Version 3.0 assessments use v3.0 schemas. The same validation engine handles all versions with just different schema configurations. Providers access validation through web interfaces or APIs. Results are consistent regardless of platform. The schema is the source of truth.

The Direct Parallel to Environmental Regulations

We built the SIF Lexer℠ before large language models existed. The pattern recognition was deterministic—engineered rules for parsing regulatory language, not neural networks. It worked because CMS specifications, like federal regulations, follow consistent linguistic structures.

Environmental regulations use identical patterns:

CFR Text: “Implementing soil control measures on existing agricultural lands, such as grade stabilization structures...”
Pattern recognized: Action category + land classification constraint + enumerated practice list

CFR Text: “Not require substantial dredging, excavation, or placement of fill”
Pattern recognized: Threshold constraint with negation

CFR Text: “Incorporate the applicable NRCS conservation practice standards as found in the Field Office Technical Guide”
Pattern recognized: External reference requiring document retrieval

The linguistic structure is there. A parser that recognizes these patterns, whether deterministic like our original Lexer or AI-powered like modern large language models, can generate executable schemas.

Modern AI doesn’t replace this architecture. It accelerates it. What the SIF Lexer℠ does through pattern-matching rules, AI does through context-aware understanding. The validation engine remains the same. The schema format remains the same. Only the extraction layer improves, from deterministic parsing to natural language comprehension.

The fundamental insight transfers directly to environmental compliance: regulations are human-readable specifications that encode machine-executable logic. Extract that logic into schemas. Execute schemas with validation engines. Update schemas when regulations change with no code deployment required.

Technical Implementation

Consider this regulatory text from NRCS environmental compliance:

> "Replacing and repairing existing culverts, grade stabilization, and water control structures and other small structures that were damaged by natural disasters where there is no new depth required and only minimal dredging, excavation, or placement of fill is required."
>
> — ***7 CFR § 1b.4(d)(6)***

This prose contains executable logic. The regulation establishes four testable conditions: structures must be pre-existing, damage must result from natural disasters, depth cannot exceed original construction, and disturbance must remain minimal.

Structured as machine-readable policy:

```
{
  "catex_id": "USDA-06d-NRCS",
  "title": "Repair of Disaster-Damaged Water Control Structures",
  "authority": "7 CFR § 1b.4(d)(6)",
  "effective_date": "2025-07-03",

  "applicability_conditions": [
    {
      "condition_id": "c001",
      "requirement": "Structures must be pre-existing",
      "data_element": "structure_status",
      "valid_values": ["existing", "previously_constructed"],
      "validation_rule": "structure_status IN valid_values"
    },
    {
      "condition_id": "c002",
      "requirement": "Damage must result from natural disaster",
      "data_element": "damage_cause",
      "valid_values": ["flood", "hurricane", "tornado", "earthquake", "wildfire"],
      "validation_rule": "damage_cause IN valid_values"
    },
    {
      "condition_id": "c003",
      "requirement": "No new depth beyond original construction",
      "data_element": "excavation_depth_feet",
      "threshold": 0,
      "validation_rule": "excavation_depth_feet <= original_depth"
    },
    {
      "condition_id": "c004",
      "requirement": "Only minimal disturbance",
      "data_element": "disturbance_volume_cubic_yards",
      "threshold": 50,
      "validation_rule": "disturbance_volume_cubic_yards < threshold"
    }
  ],

  "determination_logic": {
    "if_all_conditions_met": "Categorical Exclusion applies",
    "if_any_condition_fails": "CatEx does not apply - prepare EA or EIS"
  }
}
```

This format executes directly. A validation engine reads the schema, evaluates project data against the conditions, and returns a determination. When a culvert damaged in a declared flood requires 35 cubic yards of excavation at the original 8-foot depth, the system validates all four conditions and approves the categorical exclusion in milliseconds.

When regulations change, schemas update. No code deployment required.

Application: Post-Disaster Culvert Repair

Consider a disaster recovery project: replacing a culvert damaged by severe flooding. The NRCS field office must determine whether this qualifies for categorical exclusion.

The regulation specifies that "replacing and repairing existing culverts" damaged by "natural disasters" with "no new depth" and "only minimal dredging, excavation, or placement of fill" may qualify under 7 CFR § 1b.4(d)(6).

Each regulatory phrase becomes a testable condition:

```
{
  "project_validation": {
    "project_id": "NRCS-EWP-2026-0423",
    "project_name": "County Road 15 Culvert Replacement",

    "conditions_evaluated": [
      {
        "condition_id": "existing_structure",
        "requirement": "Structure must be pre-existing, not new construction",
        "actual_value": "existing",
        "validation": "existing IN ['existing','previously_constructed']",
        "result": "PASS"
      },
      {
        "condition_id": "disaster_damage",
        "requirement": "Damage must result from declared natural disaster",
        "actual_value": "flood",
        "validation": "flood IN ['flood','hurricane','tornado','earthquake','wildfire']",
        "result": "PASS"
      },
      {
        "condition_id": "no_new_depth",
        "requirement": "Excavation depth cannot exceed original construction",
        "actual_value": 8.0,
        "original_depth": 8.0,
        "validation": "8.0 <= 8.0",
        "result": "PASS"
      },
      {
        "condition_id": "minimal_disturbance",
        "requirement": "Disturbance volume below minimal threshold",
        "actual_value": 35,
        "threshold": 50,
        "validation": "35 < 50",
        "result": "PASS"
      }
    ],

    "final_determination": {
      "all_conditions_met": true,
      "categorical_exclusion_applies": true,
      "determination": "CatEx 7 CFR § 1b.4(d)(6) applies",
      "next_step": "Complete NEPA documentation per 7 CFR § 1b.3(g)",
      "processing_time_seconds": 3.8
    }
  }
}
```

The validation engine evaluates each condition against project data. When engineering records show the culvert existed prior to the disaster, FEMA declared a flood emergency, repair depth matches the original 8-foot specification, and excavation totals 35 cubic yards, the system returns "CatEx

applies" with complete documentation of the decision basis.

The responsible official retains decision authority. The analytical work of checking conditions, validating against standards, and documenting the determination occurs automatically. Determinations that previously required days of manual review now complete in seconds.

> "Modern AI handles regulatory language as humans do, by understanding intent and context rather than matching rigid patterns.

Capabilities Beyond Basic Validation

This white paper demonstrates the foundational architecture.

Policy version management: Running multiple versions of the same regulation concurrently, with projects automatically routed to the appropriate version based on submission date or regulatory trigger dates.

Nested conditional logic: Handling complex rule hierarchies where condition B only applies if condition A is met, or where geographic location determines which subset of requirements apply.

Threshold calculations: Managing numeric comparisons across different units of measurement, with automatic conversions and boundary condition handling.

Geographic constraint evaluation: Processing spatial rules that depend on proximity to wetlands, floodplains, critical habitat, or other location-specific features.

Consultation verification: Tracking required agency consultations (Section 106 historic preservation, Section 7 endangered species, Section 404 wetlands) and validating that appropriate concurrences have been received.

Citation trail generation: Automatically documenting the evidentiary basis for each validation decision, with references to specific pages and sections of source documents.

Multi-agency policy libraries: Maintaining schemas for different agencies' regulations in a common format, enabling cross-agency consistency checks.

The Role of AI in Acceleration

Recent advances in large language models have transformed the most time-consuming aspect of this approach: the initial extraction of structured logic from regulatory prose.

Large language models overcome a fundamental limitation of deterministic lexers. The SIF Lexer[SM] required regulatory text to follow consistent linguistic patterns—specific signal words, structured phrasing, predictable syntax. When specifications deviated from these patterns, human intervention was necessary to restructure the text or manually encode the logic.

Modern AI handles regulatory language as humans do, by understanding intent and context rather than matching rigid patterns. Specifications can be written naturally. Conditional logic doesn't require standardized phrasing. Cross-references, exceptions, and nuanced requirements that would have required manual translation now extract automatically. The architecture remains identical. The extraction layer simply became more flexible.

What previously required weeks of manual analysis—reading through CFR sections, identifying conditional statements, cataloging data requirements—now occurs in hours. An AI system reads the regulatory text and generates a draft schema. Subject matter experts review the output, make corrections, and validate against case law and agency guidance.

The extraction layer benefits from AI. The validation engine that executes the rules and generates determinations operates on deterministic logic that

has proven reliable in production environments.

When agencies require "faster NEPA compliance" or "automated categorical exclusion processing," the solution is not built from scratch. It extends a working architecture that has been executing federal policy logic in operational systems.

From Healthcare Assessments to NEPA Compliance

The SIF architecture demonstrates patterns that transfer directly to environmental compliance.

Human-readable to machine-executable transformation: The SIF Lexer℠ converts English-language specifications into JSON validation schemas without human coding. Annual CMS specification updates become executable schemas in days. The same transformation applies to categorical exclusions: CFR text analyzed by AI, validation conditions extracted, schemas generated.

Concurrent policy versions without code changes: LTCH v2.0, v3.0, and v4.0 run simultaneously. Each version has its own schema. The validation engine is version-agnostic—it executes whatever schema matches the assessment version. When environmental regulations change, add a new schema. Previous versions continue unchanged. No application redeployment.

Consistency through deterministic execution: The same validation rule appears across instrument versions under different identifiers (edit code 3932a in v2.0 is edit 3951 in v3.0). SIF eliminates this maintenance problem by separating rules from code. The schema defines validation logic. The engine applies it uniformly. For NEPA: the same CatEx condition applies identically across all state offices and all reviewers because the schema is the authoritative source.

Specification updates measured in days: What required several developers and multiple months now takes a couple of specialists verifying the schema over the course of a week. The Lexer parses updated specifications, generates schemas, automated testing validates correctness. For environmental compliance: when regulations change, AI extracts updated logic, specialists validate schemas, deployment happens without code changes. Weeks instead of months.

Non-technical stakeholder participation: Clinical experts who understand CMS requirements can verify schemas without reading code. The schema structure is readable. For NEPA: environmental specialists can review categorical exclusion schemas, validate that applicability conditions match regulatory intent, approve deployment—no developer intermediary required.

Why This Matters Now

Most policy automation initiatives focus on AI's ability to read and understand regulations. Natural language processing, semantic analysis, and large language models are technologies that dominate the conversation.

But reading regulations is not the innovation.

The innovation is the **transformation architecture**—taking human-readable policy and making it machine-executable without writing custom code for each rule.

We built that architecture before AI existed as a practical tool. The SIF Lexer℠ parses regulatory language, recognizes patterns ("must," "shall," "required when"), extracts validation logic, and generates executable schemas. Deterministic rules, not neural networks. It works because specifications follow linguistic patterns.

What changed is the **extraction layer**. The SIF

Lexer[SM] handles structured CMS specifications with predictable formats. Large language models handle unstructured CFR text with variable phrasing. But both produce the same output: JSON schemas that validation engines execute.

The hard problems remain solved: 1. **Concurrent policy versions** without code deployment (SIF runs v2.0, v3.0, v4.0 simultaneously) 2. **Consistent execution** across all users and locations (schema defines truth) 3. **Rapid updates** when regulations change (days, not months) 4. **Audit trails** showing exactly which rules applied to each determination.

SIF proved the architecture works. AI accelerates the extraction. The validation engine, the schema approach, and the deployment model remain unchanged. What we built for healthcare assessments applies directly to categorical exclusions. The domain expertise changes (NEPA specialists instead of clinical experts), but the transformation pipeline is identical: regulatory text ➡ schema generation ➡ deterministic validation.

Implications for NEPA Modernization

Environmental compliance presents an ideal application for this approach. NEPA regulations contain explicit decision trees: categorical exclusions have defined applicability conditions, environmental assessments follow structured analysis frameworks, and extraordinary circumstances trigger specific review requirements.

The CEQ NEPA Data Standard provides the common structure. Policy-as-code provides the automation mechanism. Together, they enable agencies to process environmental reviews with greater speed and consistency while maintaining complete audit trails.

A conservation practice that meets all categorical exclusion conditions can receive an automated preliminary determination. The responsible official reviews the analysis, validates the citations, and signs the determination. What changed is not the decision authority, but the analytical capacity that supports that decision.

Next Steps

If regulatory implementation creates backlogs, if consistency across offices is difficult, if policy updates take months to deploy, the architecture described here provides a solution. Not a theoretical one. A working one.

We have been deliberately selective about what this white paper reveals. The Lexer demonstrates our core innovation: converting regulatory language to executable logic without custom coding. The validation engine proves production reliability. The question is whether you want to develop this capability internally or partner with Cadmus, the organization that already built it.