SvedbergOpen
DISSEMINATIONOFKNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

The Pre-Ingress Validation Gate: A Dependency-Based Taxonomy for Enforcing Regulatory Rules at the Integration Boundary

Bhanu Teja Gadde

Independent Researcher, India.

Abstract
Regulated workflow platforms commonly validate submissions after admitting them. A record is created, assigned an identifier and an owner, and made queryable, and only then is it evaluated against the rules it must satisfy. Rejection at that point unwinds state rather than preventing it. This paper describes the alternative, pre-ingress validation, in which regulatory rules are enforced at the integration boundary and a failing submission never becomes a platform record. It makes five contributions from the author's design and implementation work on a regulated submission pipeline. Two are analytical: a taxonomy that categorizes regulatory rules by which authoritative source each rule must consult, and an enforcement-placement model derived from that dependency. Three are implemented and in production: a structured findings and error taxonomy that makes each failure individually addressable downstream; an override workflow in which a named reviewer's justification is captured at the moment of decision; and an audit design that links every override to its originating finding and case reference. The rule-evaluation logic at the boundary is a separate upstream component built by another team, and the paper distinguishes it from the author's work throughout. Five regulatory rule categories are defined, the cases that do not fit them are named, and for each category the failure that would follow from enforcing it downstream is stated. The paper compares the approach with the closest prior work on policy enforcement points, business process compliance and architecture-based compliance argumentation. No submission volumes, rejection rates or override frequencies are reported, because those figures cannot be disclosed; the production scope that can be stated is given instead. This is an architecture and design contribution, not an empirical study.

Keywords: validation architecture, integration boundary, regulatory compliance, regulatory rule taxonomy, policy enforcement point, fail-safe defaults, override authority, audit trail, compliance engineering, regulated workflow architecture.

I. Introduction

A workflow platform in a regulated domain must not act on data that the governing authority never sanctioned. The conventional way to meet that obligation is to validate inside the platform: the submission arrives, a record is created, rules run against it, and failures are surfaced for correction.

By the time those rules run, the record exists. It has an identifier, an owner, an audit trail, and a queryable presence visible to reporting and to any integration that reads the platform's data model. Rejecting it is therefore a retraction rather than a refusal, and a retraction must find and undo everything the record's brief existence touched.

Pre-ingress validation inverts this. Rules are enforced at the integration boundary and a failing submission is never admitted, so it acquires no identifier, no ownership, no audit trail and no automation surface.

The paper's central argument is that where a rule is enforced is determined by which authoritative source the rule must consult. A rule adjudicable from the payload alone can be enforced anywhere. A rule requiring live access to an external authority can be enforced correctly only where that access exists. Organizing rules by that dependency, rather than by subject matter, turns a rule catalogue into a taxonomy with architectural consequences.

The contributions, detailed in Section II, are:

1. **A dependency-based regulatory rule taxonomy** (Sections IV to VI, the author's analysis): five regulatory rule categories defined by the authoritative source each must consult.
2. **An enforcement-placement model** (Section VII, the author's analysis): for each category, where it can be enforced correctly and what fails if it is enforced downstream instead.
3. **A findings ingestion, classification and routing design** (Section IX, designed and built by the author): the path by which the boundary's structured findings are taken into the platform, classified by an error taxonomy, routed and displayed so that each is individually addressable.

4. **An override and justification workflow** (Section VIII, designed and built by the author): a named reviewer, a mandatory justification captured at the moment of decision, and a distinction between rules that admit judgement and rules that do not.
5. **An audit and traceability design** (Section VIII, designed and built by the author): every override linked to its originating finding and case reference, persisting on the record.
Section III makes the pre-ingress distinction precise. Section X sets out design implications that follow from the taxonomy. Section XI compares the approach with the closest prior work. Section XII identifies potential applications beyond this implementation. Sections XIII to XV give the evidence position, the limitations and the conclusion.

II. The author's contribution

Figure 1 shows the submission path end to end, and marks which components the author designed and built, which component is upstream and built by another team, and which systems are external authorities.

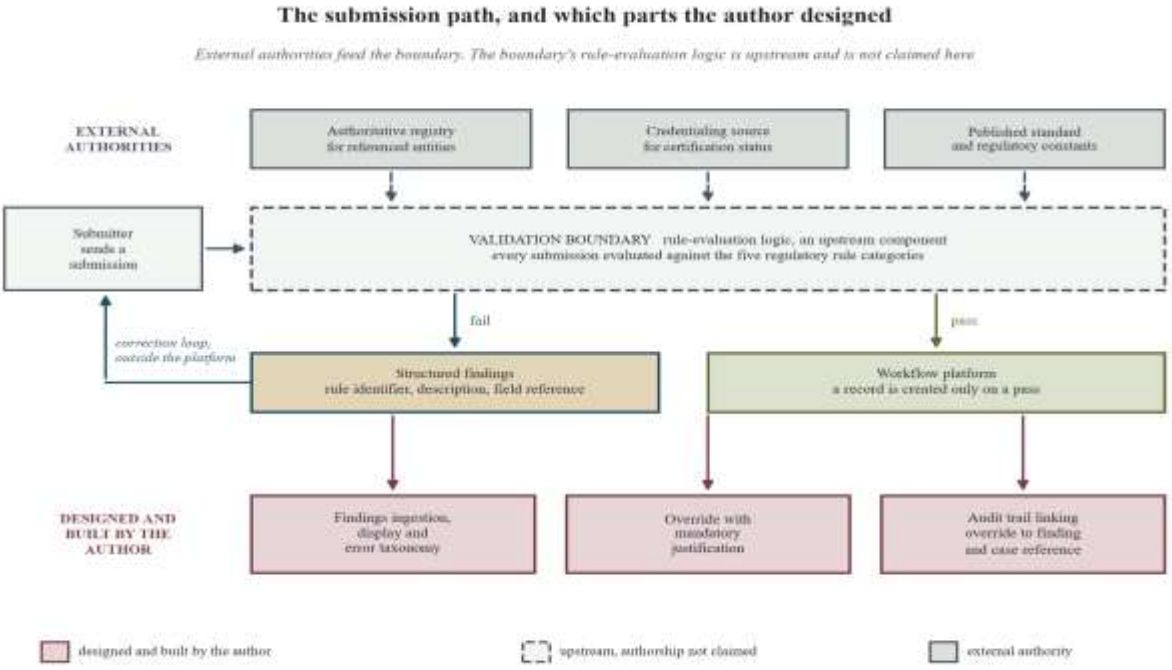


FIG. 1. The submission path. The boundary's rule-evaluation logic is an upstream component built by another team. The findings ingestion and error taxonomy, the override workflow and the audit trail are the author's design and implementation.

Table I summarises each contribution in the same terms: the problem that existed, the author's decision or analytical position, why it was necessary, what the author implemented or analysed, the technical consequence, and the evidence available for it.
The first two rows are analytical contributions and are labelled as such. They state the author's position on how rules should be categorised and where enforcement belongs. **They do not claim authorship of the production placement decision, which was made by the upstream team.** The remaining three rows are systems the author designed, built and operates.

TABLE I. The author's contributions.

Contribution	Problem	Author's decision or analytical position	Why necessary	What the author implemented or analysed	Technical consequence	Available evidence
Dependency-based rule taxonomy	The rule set was described by subject	**Analytical position:** rules should	Subject matter does not constrain	An analysis of the production rule set of	Placement becomes derivable	The analysis in this paper; not

	matter, which says nothing about where a rule can run	be categorized by the authoritative source each must consult	placement; source dependency does	more than forty rules into five categories (Sections IV to VI)	from the category rather than decided rule by rule	independently reviewed
Enforcement-placement model	No stated basis for deciding which layer enforces which rule	**Analytical position:** a rule belongs where its authority is reachable, and the rest are best kept at the same boundary for one audit model	Two categories cannot be adjudicated correctly downstream; splitting the others carries an audit cost	The placement argument of Section VII, offered as an account of why the production placement is the correct one. The placement itself was decided and implemented by the upstream team	On the controlled ingestion path, a defect in the platform's internal validation cannot admit data the upstream authority never sanctioned	The argument, and the production placement it accounts for; no comparative measurement
Findings ingestion, classification and routing	The platform received findings it could not address individually, so independently correctable problems collapsed into one rejection	Ingest each finding as an individually addressable item, classify it with an error taxonomy, and route and display it on that basis	A finding the platform cannot address individually cannot be routed, displayed or overridden individually	The findings ingestion and display path and the error taxonomy that classifies findings (Section IX). The structure of an individual finding is determined upstream at the boundary	Findings can be displayed, routed and acted on independently once they cross the boundary	In production since August 2024; finding volumes not disclosable
Override and justification workflow	An override that records only that a rule was not enforced leaves no accountable decision	Require a named reviewer and a mandatory justification captured at the moment of decision	Reasoning reconstructed afterwards is reconstruction; the moment of decision is the only time the actual reason exists	The override workflow for the three overridable conditions (Section VIII)	Every override carries who decided and why, so the override path does not weaken the gate	In production; override frequency not disclosable
Audit and traceability design	An override was not connected to the finding that produced it	Link each override to its originating finding and case reference, and persist both on the record	An audit that cannot reach from a decision back to its finding cannot reconstruct why the decision was made	The audit trail (Section VIII)	The record shows which rule was overridden, on whose authority, and why	In production; not independently audited

III. What pre-ingress means

Pre-ingress is not a scheduling choice. **In-platform validation admits the submission into the platform's data model and then evaluates it**: between admission and rejection the record is real, and rejection must unwind that state. **Pre-ingress validation evaluates the submission at the integration boundary and admits it only on success**: a failing submission produces a structured rejection and nothing else. Figure 2 sets the two arrangements side by side.

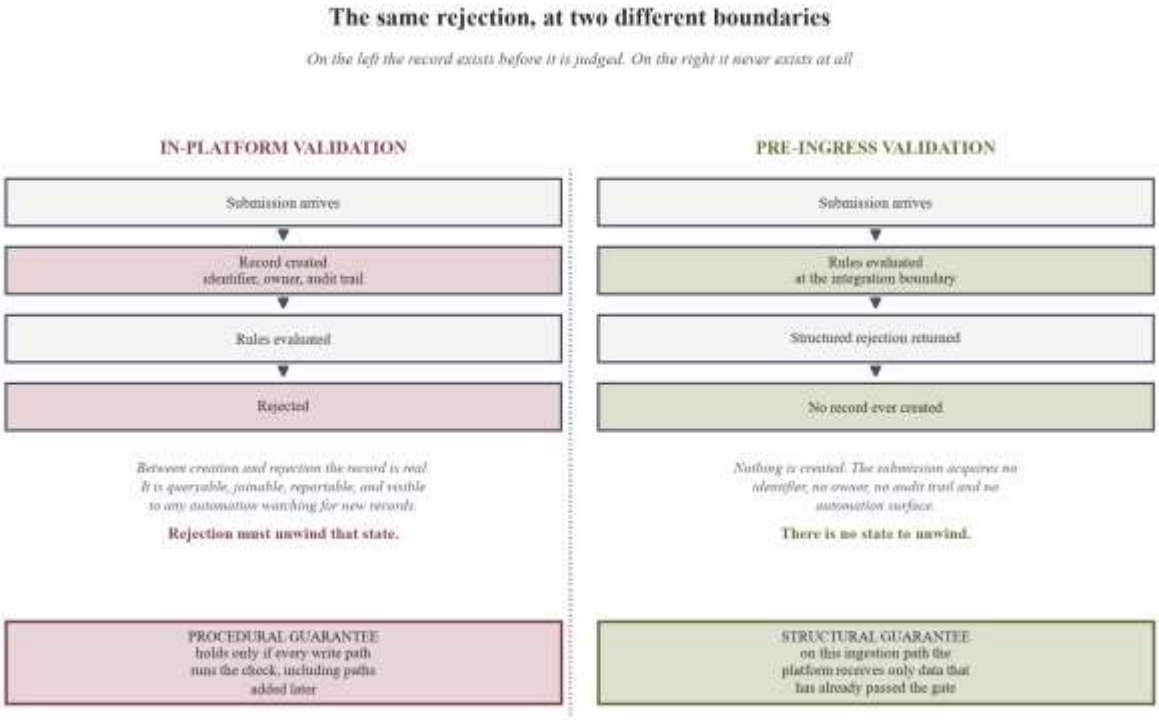


FIG. 2. The same rejection, at two different boundaries. On the left the record exists before it is judged; on the right it never exists at all.

The difference is in the kind of guarantee available. In-platform validation offers a procedural guarantee: the controls catch unauthorized data provided every code path that creates a record runs them. That guarantee is a property of the code rather than of the architecture, so it weakens as the code grows. A new object type, a bulk import written for a migration, an integration built by another team, or a path that predates the control can each create a record without the check firing.

Pre-ingress validation offers a structural guarantee, and it should be stated at the strength the architecture supports. **On the controlled ingestion path, the platform receives only submissions that have already passed the gate**, because the boundary sits between the submitter and the platform. The guarantee is conditional on one architectural commitment: that this boundary is the only route by which submissions of this kind enter. That commitment is enforceable at the integration layer and auditable from outside, which makes it stronger than a control embedded in application code. It is not a claim that the platform cannot hold unauthorized data by any route whatsoever.

This applies the principle of fail-safe defaults and complete mediation to compliance content rather than to access [1]: the safe state is the default, and every request on this path passes the check because there is no route around it. It is also an instance of the end-to-end argument [2], which holds that a function should be implemented at the layer possessing the knowledge required to implement it correctly. The rules in this domain require live access to external authorities, and the boundary is the layer where that access exists.

IV. The classification principle

The usual way to organize a rule catalogue is by subject matter: value rules, address rules, credential rules. That organization is easy to read but says nothing about where a rule can be enforced.

The principle proposed here, and applied by the author to the production rule set, is that a rule's category is determined by which authoritative source it must consult in order to be adjudicated. That dependency determines which system layer is capable of enforcing the rule correctly.

The principle produces a spectrum. At one end are rules adjudicable from the payload alone, requiring no external access: a rule that checks whether an identifier is internally well-formed can be evaluated by anything holding the identifier. At the other end are rules whose adjudication requires a live query to a specific external registry, and which cannot be evaluated correctly by any component lacking that access. Between them are rules requiring a fixed published reference but not a live lookup.

What separates the categories is dependency, not topic. Two rules about the same business object belong to different categories if one is self-adjudicating and the other requires a registry query, and two rules about different objects belong to the same category if they share a dependency profile.

V. The five regulatory rule categories

The problem the taxonomy addresses is that a subject-matter catalogue gives no basis for placing a rule. **The decision** was to categorize by dependency, and **the result**, from the author's analysis of the production rule set, is five categories. Figure 3 orders them by what each must consult, and Table II defines them with an example condition for each.

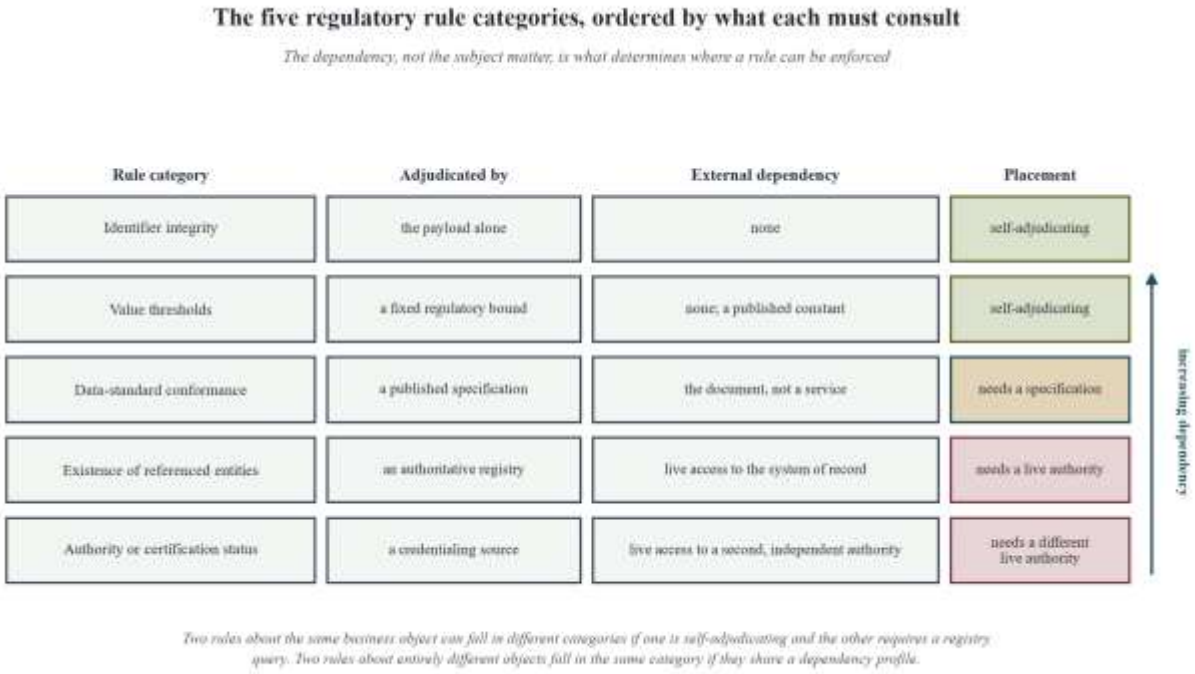


FIG. 3. The five regulatory rule categories ordered by what each must consult. The dependency, not the subject matter, determines where a rule can be enforced.

TABLE II. Regulatory rule categories by authoritative-source dependency.

Rule category	Adjudicated by	External dependency	Example condition
Identifier integrity	Self-consistency of the payload alone	None	A case number is structurally well-formed. The governing programme publishes a fixed ten-digit structure whose segments carry defined meanings and whose final position is a check digit [3]

Existence of referenced entities	A query to an authoritative registry	Live access to the system of record	A subject property resolves to a real, verifiable address
Authority or certification status	A query to a separate credentialing source	Live access to a second, independent authority	The submitting appraiser appears on the official roster
Value thresholds	Comparison against a fixed regulatory bound	None; the bound is a published constant	An appraised value is not below the regulatory floor
Data-standard conformance	Comparison against a published specification	The specification, not a live service	The submission conforms to the governing data standard

Three properties of this table carry the argument.

Identifier-integrity and value-threshold rules are self-adjudicating. They require nothing beyond the payload and a constant. They could be enforced anywhere, including inside the platform, and their placement at the boundary is a choice justified in Section VII.

Entity-existence and certification-status rules are not self-adjudicating, and they depend on different authorities. Entity existence is adjudicated by the system of record for the entity. Certification status is adjudicated by a credentialing body with no relationship to the case: an appraiser's roster status is a fact about the appraiser, independent of any particular submission. **These are two distinct dependencies.** Collapsing them, as a subject-matter scheme would by grouping both under "external checks", loses the fact that a system can have access to one authority and not the other.

Data-standard conformance depends on a specification rather than a service. A published data standard is a document, so conformance can be evaluated by anything holding it. That makes conformance checking structurally closer to identifier integrity and value thresholds than its apparent complexity suggests, and it is why conformance checking can be pushed to the submitter's own tooling in a way that a roster check cannot.

The consequence is that placement can be read from the category rather than decided rule by rule, which Section VII develops and Section X turns into a procedure for new rules.

VI. What the categories do not cover

A taxonomy is judged on completeness and mutual exclusivity. In the author's analysis, three things in this domain resist the five categories. Figure 4 places them relative to the categories, and Table III states what each one does to the scheme.

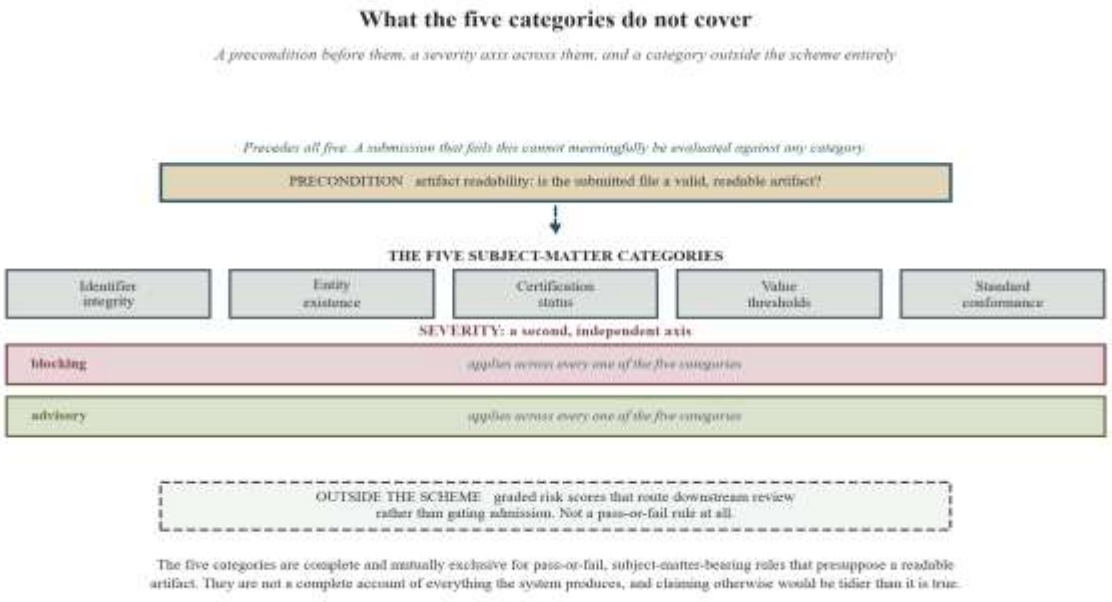


FIG. 4. The five regulatory rule categories, the severity axis that cuts across all of them, and the two things that sit outside the scheme entirely.

TABLE III. What does not fit, and why.

Residual	Nature of the problem	Consequence for the taxonomy
Artifact readability	A check on whether the submitted file is itself a readable, valid artifact	**Precedes** all five categories. A submission whose file integrity fails cannot meaningfully be evaluated for identifier, entity, authority, value or standard conformance at all
Blocking versus advisory severity	Findings are split into a group that blocks admission and one that does not, despite sharing the same reporting shape	**Cuts across** all five categories rather than sitting inside any one. It is a severity distinction, not a subject-matter one, so the scheme needs a second, independent axis
Graded risk scores	Continuous scores that feed downstream review routing rather than gating admission	**Sits outside** the pass-or-fail framework entirely. It is not a rule in the sense the categories use the term

Artifact readability is a precondition, not a category. It establishes whether the other checks are meaningful, so treating it as a sixth category would imply it is one option among peers when it gates the applicability of all of them.

Severity is a second axis, and the taxonomy is two-dimensional. A rule has a subject-matter category and a severity, and the two vary independently: an entity-existence finding may block or merely advise, and so may a data-standard conformance finding. The successor data standard for this domain supports treating severity as its own axis. It abandons the two-valued blocking and non-blocking split and replaces it with findings carrying one of four graded severities [4], which is the same axis given more resolution.

Graded scores are a different object. A pass-or-fail rule and a continuous risk score both produce findings, but only one can gate admission.

The five categories are therefore complete and mutually exclusive for pass-or-fail, subject-matter-bearing rules that presuppose a readable artifact, and they are not a complete account of everything the system produces. Two of the three residuals indicate the taxonomy's natural extension: a precondition stage before the categories, and severity as an orthogonal axis across them.

VII. Enforcement placement and the failure it prevents

The problem is that without a stated basis, which layer enforces which rule is decided case by case, and the decision is invisible until something is admitted that should not have been. **The author's decision** was to derive placement from the category. Table IV gives the placement for each category and the failure mode of enforcing it downstream instead.

TABLE IV. Enforcement placement by rule category, and the failure mode of enforcing downstream instead.

Rule category	Enforced at	Why there	What breaks if enforced in the workflow layer
Existence of referenced entities	Integration boundary	Requires live, trusted access to the system of record	Either a cached or replicated copy is consulted, weakening freshness, or a live callback is made from inside the platform back to the system of record, which reconstructs the dependency the boundary exists to remove and exposes the platform to that dependency's failure modes
Authority or certification status	Integration boundary	The credentialing source is a second, independent authority	The platform can hold a record before ever confirming its submitter was authorized to produce it
Identifier integrity	Integration boundary	Adjudicable anywhere; placed here for a single audit model	Compliance surface splits across two enforcement points with two different failure profiles

Value thresholds	Integration boundary	Adjudicable anywhere; placed here for a single audit model	As above
Data-standard conformance	Integration boundary	Adjudicable anywhere; placed here for a single audit model	As above

Why the decision was necessary rests on two different arguments, which should not be conflated.
For entity existence and certification status the argument is necessity. These rules cannot be adjudicated correctly downstream without either degrading the check or rebuilding the dependency. A cached roster is a roster as it was, not as it is, and certification status is exactly the kind of fact that changes between caching and use. A live callback from inside the platform is worse: it makes the platform's ability to admit a record contingent on an external service being reachable, which is the coupling the boundary was introduced to remove.
For identifier integrity, value thresholds and data-standard conformance the argument is coherence, not necessity. These could be enforced downstream without loss of correctness. Placing them at the same boundary keeps the compliance surface under one audit model rather than splitting it across two enforcement points with different failure characteristics, different logging and different owners. **This is a weaker argument than the one for entity existence and certification status, and it is stated as one.**
What it corresponds to in practice. The production boundary places all five categories at the integration boundary, and the placement model is the author's account of why that placement is correct for two categories and justified for the other three.

A. The consequence: the fail-open condition is closed on this path

The failure that boundary enforcement prevents on the controlled ingestion path is worth stating precisely, because it is silent. If one of these categories is checked downstream instead, then a defect in the platform's internal validation logic, a code path that bypasses the check, or an omission when a new object type is added, allows data the upstream authority never sanctioned to persist as a real, queryable record. **Nothing surfaces.** The record looks like every other record, and its audit trail records that it was created, not that it was never authorized.
This is a fail-open condition, the failure mode that security architecture has treated as unacceptable in mediation since the principle of fail-safe defaults was articulated [1]. A validation control that fails by admitting is worse than one that fails by rejecting, because rejection is visible and admission is not.
It also explains why the robustness principle, which counsels being liberal in what one accepts [5], **is the wrong guidance here and is cited as the position this design rejects.** Liberality in acceptance is appropriate where the cost of rejecting valid input exceeds the cost of accepting invalid input. In regulatory gating the ordering is reversed: an unauthorized record that persists is a compliance failure, while a rejected submission is a correction loop.

VIII. Override and the justification workflow

Override is often treated as an operational escape hatch added after the rules are written. In this design it is part of the scheme, and the override workflow and audit trail described in this section are the author's own implementation. Figure 5 draws the two paths a failing rule can take.

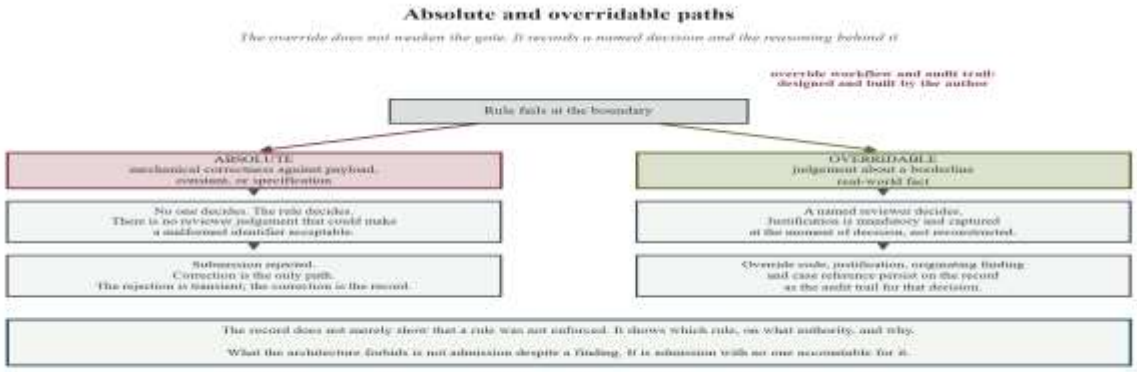


FIG. 5. Absolute and overridable paths. The override workflow and audit trail are the author's implementation. The override does not weaken the gate: it records a named decision and the reasoning behind it, and the record carries both forward.

A. Which rules admit override

Of the rule conditions this system exposes, three are overridable and the remainder are absolute: an appraised value below the published floor, an unresolved subject address, and appraiser roster membership. Each is marked overridable in the governing programme's published hard-stop list [3]. That list marks a larger set of conditions overridable than the three handled here, and a considerably larger set as not overridable; the three above are the subset within this system's scope.

The published definition is worth stating precisely, because it is not what an outsider would assume. A non-overridable check indicates an error that must be resolved before a submission can be transmitted successfully. An overridable check indicates a possible discrepancy that the submitter may need to address but which **does not need to be corrected for the transmission to succeed** [3]. An override is therefore not an escalation that unblocks a blocked submission; it is the recorded disposition of a flag that was never going to block on its own. What the override supplies is not permission but attribution.

That separates two properties that are easily confused: overridability describes who may dispose of a finding, while severity, the axis identified in Section VI, describes whether the finding gates admission at all.

The published list supplies direct evidence for that separation, and it is worth drawing out because it is the strongest external support the paper has for the second axis. Within the authority and certification category, roster membership is overridable while the conditions covering certification validity, active status and eligibility at the appraisal date are not. Within the entity existence category, an unknown or invalid subject address is overridable while a missing one is not. **In both cases the category is constant and the overridability differs**, which is precisely what a scheme that inferred severity from category would get wrong. The dependency taxonomy determines where a rule can be enforced; it does not determine how a failure is disposed of, and the published list confirms that these vary independently.

The author's analysis of the three overridable conditions is that each turns on a judgement about a borderline real-world fact rather than on a mechanical correctness check. Whether a property address resolves is a question about the world, and address data is imperfect in ways a reviewer can sometimes adjudicate and an algorithm cannot. Whether an unusually low valuation is an error or a genuine reflection of the asset is similarly a judgement. **The mechanically adjudicable categories are the ones that are absolute**: an identifier is well-formed or it is not, and no reviewer judgement could make a malformed identifier acceptable.

B. The workflow the author built

The problem. An override that records only that a rule was not enforced leaves an unaccountable gap in the compliance record. Someone allowed a submission through a flagged condition, and neither the identity of that person nor the reason survives.

The author's decision. Every override is paired with a mandatory justification field, and the pairing is enforced by the workflow rather than left to reviewer discipline. An override cannot be recorded without a reviewer's stated reasoning captured alongside it at the moment of decision.

Why it was necessary. Reasoning reconstructed after the fact is reconstruction. The moment of decision is the only time the actual reason is available, which is why the field is mandatory at that moment rather than a later annotation.

What the author implemented. The override workflow for the three overridable conditions, and an audit trail that persists the override code, the justification, the originating finding and the case reference on the record.

The consequence. The record does not merely show that a rule was not enforced. It shows which rule, on whose authority, and why. This is what makes override compatible with the guarantee in Section III: the gate is not weakened by the existence of an override path, because the override is a recorded, attributable act rather than an absence of enforcement. The published federal guidance for this domain draws the same distinction, describing some checks as requiring correction before a submission can proceed and others as clearable through a user-controlled override process [3].

C. The mechanism is being retired

The override model described above is specific to the current data standard, and that standard is being replaced. The successor removes manual hard-stop overrides entirely, retiring both the override action and the Client Override User role that held it, on the basis that revised validation and submission processing methods supersede the need for manual overrides [4]. In their place the successor introduces a graded severity model in which only the most severe level blocks a submission and the remaining levels do not affect its status [4], which is the successor to the overridable class rather than its abolition. The governing agency has stated that its adoption dates are still to be announced [6], so the timing is open but the direction is settled.

This is a genuine limitation on the override contribution. A design built around a mechanism scheduled for removal has a bounded operational life. The design principle does not depend on the mechanism: rules turning on a borderline judgement about the world need a named human disposition with recorded reasoning, and

mechanically adjudicable rules do not. What the successor standard changes is where that judgement is expressed, not whether it is required.

IX. Structured findings and the correction loop

A submission failing a pre-ingress rule is rejected with a structured response identifying **which specific rule or rules failed**, rather than a single undifferentiated rejection. The findings path and the error taxonomy that classifies its output are the author's implementation.

The problem. A single undifferentiated rejection tells the submitter that something is wrong without saying what, forcing a diagnostic loop the gate already has the information to eliminate. It also collapses independently correctable problems into one, so a submission with three faults is corrected and rejected three times.

The author's decision. Represent each failure as an individually addressable finding carrying a rule identifier, a description, and a reference back to the specific document or field involved, and classify findings with an error taxonomy that the downstream workflow consumes.

Why it was necessary. The workflow platform must act on findings, not on rejections. A finding it cannot address individually cannot be routed, displayed or overridden individually.

What the author implemented. The findings ingestion and display path, and the error taxonomy.

The consequence. The workflow platform can display, route and act on distinct findings independently once they cross the boundary, which is why the rejection payload is part of the architecture rather than an error-handling detail.

A. Which findings the platform ingests, and which do not reach it

The distinction matters because the platform both refuses submissions and ingests findings, and those are two different populations of finding.

Findings that block admission do not reach the platform. A submission failing an absolute rule is refused at the boundary. Its findings are returned to the submitter, who corrects the submission and resubmits. The platform has no record of the attempt, and **the correction loop for a refused submission is therefore external to the platform**: its record of the case begins with the first submission that satisfied the gate.

Findings on admitted submissions are ingested. A submission can be admitted while still carrying findings, because not every finding gates admission. The overridable conditions of Section VIII are the clearest case: they are raised as findings but do not, on their own, prevent transmission. These are the findings the ingestion, classification, routing and display path exists to handle. They arrive with the admitted submission, are classified by the error taxonomy, and are routed for disposition, which for an overridable condition means the named-reviewer override workflow and its audit trail.

The two populations are therefore complementary rather than contradictory. The boundary decides admission; the platform handles what accompanies an admitted submission. Nothing the platform ingests could have blocked the submission it arrived with.

B. The rejection as a contract

This is a contract between two systems in the sense of design by contract [7]: the boundary publishes what it requires, the submitter is obliged to meet it, and the platform is entitled to assume it. The structured rejection is what makes the obligation actionable rather than merely stated.

X. Design implications of the taxonomy

This section sets out consequences that follow from the taxonomy for anyone designing or changing a pre-ingress boundary. It is analysis rather than a description of the production system, and it does not report how the production boundary handles each case.

A. Placing a new rule

The taxonomy turns placement from a case-by-case judgement into a procedure. For a rule being added to the boundary:

1. Identify the authoritative source the rule must consult to be adjudicated: the payload alone, a published constant, a published specification, a system of record, or an independent credentialing source.
2. If the source is a live authority, the rule belongs at the boundary, because that is the only layer where the authority is reachable without rebuilding the dependency inside the platform.
3. If the source is the payload, a constant or a specification, the rule could run anywhere; placing it at the boundary is a decision for audit coherence and should be recorded as such.
4. Decide overridability separately, by asking whether the rule turns on a borderline real-world fact. If it does, route failures to the override workflow with a mandatory justification; if not, the rule is absolute.
5. Assign severity independently of the category, as the second axis of Section VI.
6. Define the finding the rule emits: its identifier, its description and the field it references, so that the downstream workflow can address it individually.

B. When an authority cannot be reached

Placing live-authority rules at the boundary makes admission depend on those authorities being reachable at the moment of submission. The taxonomy makes the resulting trade-off explicit per category. For the three self-adjudicating or specification-based categories the question does not arise, because no live service is consulted. For entity existence and certification status, the boundary must either hold the submission and return a finding stating that the authority could not be consulted, which fails closed, or admit it and flag the condition, which fails open. Consistent with the argument in Section VII.A, failing closed is the conservative default, and its cost is availability coupling: an authority outage delays submissions rather than admitting unverified ones.

C. Caching

A cache of an authority's answers trades freshness for availability. The taxonomy indicates where that trade is safe. A published specification or constant changes on a known schedule and can be cached until the next published revision. Certification status can change between caching and use, so a cached roster weakens exactly the check the category exists to perform. Entity existence sits between the two, depending on how often the system of record changes the entities concerned.

D. A worked application of the procedure

The procedure of Section X.A is only useful if it discriminates, meaning that it sends different rules to different places for stated reasons rather than confirming whatever placement already exists. Table V applies its six steps to four rules drawn from the categories of Section V. **This is an illustrative walkthrough, not an empirical evaluation**: it demonstrates that the procedure is decidable and that it separates rules, and it does not measure any outcome.

TABLE V. The placement procedure of Section X.A applied to four rules.

Step	Case number structure	Appraised value floor	Subject address resolves	Appraiser roster status
1. Authoritative source	The payload alone	A published constant	The system of record for the property	An independent credentialing source
2. Live authority?	No	No	**Yes**	**Yes**
3. Placement	Could run anywhere; at the boundary for audit coherence	Could run anywhere; at the boundary for audit coherence	**Boundary only**	**Boundary only**
4. Overridable?	No. A malformed identifier is mechanically wrong and no judgement makes it acceptable	**Yes.** A low valuation may be an error or a true reflection of the asset	**Yes.** Address data is imperfect in ways a reviewer can sometimes adjudicate	**Yes.** Roster status at the moment of submission can be disputed on grounds a reviewer can assess
5. Severity	Absolute	Flag, not a gate	Flag, not a gate	Flag, not a gate
6. Finding emitted	Rule identifier, description, field reference to the case number field	As above, referencing the valuation field	As above, referencing the address fields	As above, referencing the appraiser identifier

Two things are visible in the table that the categories alone do not show.

The procedure separates placement from overridability, and they do not correlate. The two rules that must run at the boundary are both overridable. The rule that could run anywhere is the one that is absolute. A scheme that assumed strict rules belong at the boundary and lenient ones inside would place all four wrongly.

Steps 2 and 3 do the discriminating work; steps 4 and 5 are independent. Only the third and fourth columns are forced to the boundary, and they are forced for different reasons, consulting different authorities. That is the distinction Section V argues a subject-matter catalogue loses.

E. What the failure modes show when read together

Table IV gives the failure mode of enforcing each category downstream. Read down that column rather than across it, a second qualitative result appears, and it is the one that justifies the taxonomy rather than a blanket preference for validating early.

The failures are of two different kinds, not two degrees of the same kind. For identifier integrity, value thresholds and data-standard conformance, enforcing downstream is still correct. The rule can be adjudicated there, because the payload, the constant and the specification are all equally available inside the platform. What it costs is that a record exists before it is validated, so rejection becomes retraction, and the compliance surface splits across two enforcement points. Those are engineering and audit costs, and a team that accepted them would get right answers more expensively.

For existence of referenced entities and authority or certification status, enforcing downstream is not merely more expensive. It is **incorrect**, and no downstream engineering recovers it. The platform either reconstructs the dependency the boundary exists to remove, or it admits a record the authority never confirmed. The certification case is the sharper of the two, because status can change between submission and any later check, so a record can be admitted on the authority of a submitter who was not authorized at the moment that mattered.

A uniform rule cannot express that. "Validate at the boundary" and "validate in the platform" both treat the five categories alike, and the failure analysis says three of them tolerate either placement while two do not. That asymmetry is what the dependency taxonomy captures and what a subject-matter catalogue cannot, since subject matter is silent on which authority a rule consults.

Neither this comparison nor the walkthrough in Section X.D reports a measurement. Both are analysis, and Section XIII records them as such.

F. When the data standard changes

Because categories are defined by dependency rather than by the content of individual rules, a change of data standard is expected to affect mainly the data-standard conformance category and the severity axis, not the dependency structure. The successor standard for this domain is consistent with that expectation: its principal changes are to severity grading and the removal of manual overrides [4], while the authorities that entity-existence and certification-status rules must consult are unchanged. This is an expectation derived from the taxonomy, not a measured result.

XI. Related work, the gap, and what this paper adds

A. What existing work covers

Table VI lists the nearest existing bodies of work and what each leaves uncovered.

TABLE VI. The nearest existing bodies of work, and what each does not cover.

Body of work	What it establishes	What it does not cover
Enterprise integration patterns [8]	A message-validator step in an integration pipeline	Validation is treated as undifferentiated. No categorization of validation types by dependency, and no override semantics
Policy enforcement point models [9], [10]	Formalizes where an enforcement decision is made relative to a protected resource, which is the closest existing framing to enforcing at a boundary	Addresses authentication and authorization almost exclusively. The question is whether the requester may act, not whether the content submitted is compliant
Attribute-based access control and XACML [11]	A mature separation of decision point from enforcement point, with attribute-driven policies	The policy subject is access, not regulatory compliance of the payload
API gateway and input validation work [12]	Enforcement at a gateway, and the consequences of failing to validate input	Oriented toward schema conformance and security vulnerability rather than regulatory rules, and carries no override or audit semantics
Contract-first API design and consumer-driven contracts [13]	A published, enforceable contract between producer and consumer	Concerns interface shape and compatibility, not regulatory content, and has no notion of a rule that may be overridden by a named authority

Business process compliance [14], [15], [16]	Formal modelling of control objectives and compliance rules over process models	Operates at the level of process semantics rather than architectural placement. It asks whether a process satisfies a rule, not which layer can enforce it
Compliance by design and RegTech [17], [18]	Embedding regulatory requirements into system design as a methodology	Stays at the level of principle. No per-category placement argument tied to source dependency
Architecture-based compliance argumentation [19]	Explicitly connects software architecture to regulatory compliance argument	The closest prior work. Concerns constructing an argument that an architecture complies, rather than a placement rule derived from which authority each requirement must consult
Requirements engineering for regulatory compliance [20]	Systematic treatment of eliciting and tracing regulatory requirements	Concerns the requirements lifecycle rather than runtime enforcement placement

B. The gap that remains

Each element of this paper exists somewhere in that literature. Enforcement at a boundary is the policy enforcement point model. Rule categorization exists in business process compliance. Override with justification exists in operational practice throughout regulated industries. **What the author has not located is the combination**: a rule taxonomy organized by authoritative-source dependency, an explicit argument connecting that dependency to enforcement placement, and override and audit semantics treated as part of the taxonomy rather than as operational overhead.

C. How the approach differs from the three closest bodies of work

Table VII compares the approach with the three closest bodies of work on four dimensions: the question each answers, its unit of analysis, what determines where enforcement happens, and how it treats override.

TABLE VII. Point-by-point comparison with the closest prior work.

Dimension	**Policy enforcement point models [9], [10]**	**Business process compliance [14], [16]**	**Architecture-based compliance argumentation [19]**	**This paper**
Question answered	May this requester perform this action on this resource?	Does this process model satisfy these compliance rules?	Can it be argued that this architecture complies with this regulation?	Which layer can correctly enforce this regulatory rule?
Unit of analysis	An access request	A process model and its control objectives	An architecture and the requirements argued against it	An individual regulatory rule and the authority it must consult
What determines the enforcement location	The position of the protected resource and its decision point	Not a placement question	The architecture under evaluation, taken as given	The authoritative source each rule depends on
Treatment of override	Not addressed	Not the focus	Not the focus	Part of the taxonomy, with mandatory justification and audit linkage

Against policy enforcement point models, the difference is the subject of the rule. Those models place a decision point relative to a protected resource and ask whether a requester may act. The rules here concern whether submitted content satisfies a regulation, and the placement question is which layer can adjudicate the rule at all, which depends on the rule's source rather than on the resource being protected.

Against business process compliance, the difference is the level of analysis. That work checks whether a process, as modelled, satisfies compliance rules. It does not ask where in a system's architecture each rule can be enforced, and a process can satisfy every rule on paper while its enforcement sits in a layer that cannot adjudicate some of them correctly.

Against architecture-based compliance argumentation, the closest work, the difference is direction. That work constructs an argument that a given architecture complies. This paper proposes a rule for deciding where enforcement belongs before the architecture is fixed, derived from a property of each requirement, namely which authority it must consult. The two are complementary: a placement derived from the taxonomy is the kind of design decision such an argument would then have to justify.

D. What is not claimed

An absence of located prior work is not a proof of absence, and the literature search was not a systematic review. The paper does not claim to be first, unique or without precedent. A reader assessing novelty should consult the architecture-based compliance argumentation work [19] directly and weigh the distinction drawn above rather than take it on assertion.

XII. Potential applications beyond this implementation

This section describes potential applications, not demonstrated results. The taxonomy was derived from one regulated submission pipeline, and the author has not built or evaluated a system in any of the domains below.

The transferable principle is: **for each regulatory rule, identify the authoritative source that must be consulted to adjudicate it, and place enforcement at the layer where that source is reachable and auditable.** Applying it elsewhere would require three things of a domain: rules imposed by an authority outside the system, at least one rule whose adjudication depends on a source the processing system does not own, and a boundary through which submissions enter.

Domains that appear to meet those conditions include clinical data submission, where certification and registry lookups sit outside the receiving system; financial transaction reporting, where reference data and entity identifiers are maintained by external authorities; and supply-chain compliance, where certificates of origin and conformity are issued by third parties. In each, the same question could be asked: which rules can be adjudicated from the payload, which need a published reference, and which need a live authority.

What might transfer is the question, not the five categories. Another domain may produce a different number of categories. Whether the principle produces a useful taxonomy elsewhere is untested, and Section XIV records that as a limitation.

XIII. Evidence status

Table VIII gives the standing of every claim the paper makes, so a reader can separate the analytical contributions, the implemented designs, the stated production scope and what was not measured.

TABLE VIII. Claim by claim.

Claim	Status
The classification principle and the five regulatory rule categories (Sections IV and V)	**Original analytical contribution** of the author
The completeness analysis (Section VI)	**Original analytical contribution** of the author
The enforcement-placement model (Section VII)	**Original analytical contribution** of the author
The design implications (Section X)	**Analysis** derived from the taxonomy; not a description of the production system
The override and justification workflow (Section VIII)	**Designed and built by the author**, in production since August 2024
The structured findings path and error taxonomy (Section IX)	**Designed and built by the author**, in production
The audit and traceability design (Section VIII)	**Designed and built by the author**, in production
The rule-evaluation logic at the boundary	**Built by another team** as a separate upstream component. Analysed here for its structure and placement; authorship not claimed
Production scope: one regulated submission programme, a rule set of more than forty distinct	**Stated production scope**, as reported by the author

rules, three overridable conditions, operating since August 2024	
Submission volumes, finding volumes, rejection rates by category, override frequency	**Not reported**; cannot be disclosed
Reduction in downstream failures after moving validation to the boundary	**Not measured**; no before-and-after comparison exists
Comparative evaluation against in-platform validation	**Not conducted**
Applications in other domains (Section XII)	**Potential applications**; not demonstrated
Independent technical review of this analysis	**Not obtained**

The architectural claims rest on the design and operation of the systems described and on the reasoning in Section VII, not on production statistics. A reader should treat the placement argument as an argument, and the fail-open condition in Section VII.A as a structural analysis rather than an observed failure rate.

XIV. Limitations

No empirical validation. The central claim, that boundary enforcement prevents a kind of failure that in-platform enforcement permits, is argued structurally and has not been measured.

No comparative evaluation. No study compares two systems differing only in enforcement placement. Such a study would test the central claim directly and has not been conducted.

No disclosable metrics. Submission, finding and override volumes, rejection rates, and any measure of downstream remediation cannot be disclosed, so the paper cannot show the scale of the effect it describes.

A single domain and a single system. The taxonomy was derived from one regulated submission pipeline. Whether the five categories generalize is untested; Section XII states the principle in domain-neutral terms so that it can be tested elsewhere.

The taxonomy is incomplete by construction. Section VI names three residuals. Two of them, the precondition stage and the severity axis, indicate the taxonomy's natural extension and are not developed here.

The coherence argument is weaker than the necessity argument. The justification in Section VII for placing identifier-integrity, value-threshold and data-standard conformance rules at the boundary rests on audit-model uniformity rather than on correctness, and a reader who weighs the cost of boundary enforcement differently could reasonably place those three elsewhere.

The literature search was not systematic. Section XI states this directly. A systematic review could locate prior work the author did not find.

The override analysis rests on three conditions. The association between overridability and mechanical adjudicability is drawn from three overridable rules against a larger absolute set. It is a pattern consistent with the evidence available, not a demonstrated law.

The override mechanism has a bounded life. As Section VIII.C sets out, the successor data standard retires manual hard-stop overrides [4], and the adopting agency has not announced its transition dates [6].

No independent technical review has been obtained. The originality and significance of the contribution have not been assessed by an expert other than the author.

Client-specific rules are excluded, so the paper describes the shape of the pattern rather than a reproducible rule catalogue. The public federal guidance for this domain [3] is the source for the concrete rule set.

XV. Conclusion

Where a regulatory rule is enforced is not an implementation detail. A rule that must consult an external authority can be enforced correctly only where that authority is reachable, and a rule enforced after a record exists can only retract state rather than prevent it.

The taxonomy proposed here organizes rules by that dependency: identifier integrity and value thresholds adjudicable from the payload and a constant, entity existence requiring the system of record, certification status requiring a separate credentialing authority, and standard conformance requiring a published specification. The categories differ in what they must consult, and that difference determines where each can be enforced.

The architectural consequence is that enforcement belongs at the integration boundary, before a submission enters the platform's data model. On the controlled ingestion path the platform then receives only data that has already passed, and its compliance property becomes structural rather than procedural.

The author's implemented work makes the resulting decisions individually addressable and individually accountable: a findings path and error taxonomy that turn each failure into a finding the workflow can act on, an override workflow that captures a named reviewer's justification at the moment of decision, and an audit trail that links every override to the finding that raised it.

None of this has been measured, and the paper's claims are analytical and descriptive. What it offers is a dependency-based taxonomy of regulatory rules, a placement model derived from it, design implications for anyone building such a boundary, and three implemented designs, with the limits of each stated.

Note on scope and evidence

What the author designed and built. Since August 2024 the author has designed, developed and operated the downstream systems that consume what the validation boundary produces: the ingestion, classification, routing and display of the structured findings the boundary emits, the error taxonomy that classifies them, the override workflow for the three overridable conditions discussed in Section VIII, and the audit trail that links each override to its originating finding and case reference. The structure of an individual finding is determined upstream, at the boundary; the author's contribution is everything the platform does with a finding once it crosses. All of it is in production.

What the author analysed. The dependency-based taxonomy (Sections IV to VI), the enforcement-placement model (Section VII) and the design implications (Section X) are the author's analysis of the pipeline's rule set and boundary.

What another team built. The rule-evaluation logic at the boundary itself is a separate upstream component developed by another team. This paper analyses its structure and argues for its placement; it does not claim authorship of it, and no contribution here depends on having written it. The rule set it enforces comprises more than forty distinct rules covering appraised value, appraiser certification, subject address, data-standard conformance and case-number integrity.

What is not reported. There are no submission volumes, no rejection rates by category, no override frequencies and no measurement of downstream failure reduction, because those figures cannot be disclosed. Section XIII records this claim by claim and states the production scope that can be given in their place. No internal system names or client-specific rules are disclosed. The regulatory programme itself is identifiable from the published sources cited here, which is unavoidable: the argument depends on a rule set whose governing documents are public, and citing them is what allows a reader to check the claims about overridability in Section VIII.

Field. The paper is a contribution to compliance engineering and regulated workflow architecture: the design of the integration layer at which regulatory rules are enforced on data entering an enterprise workflow platform.

References

[1] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308, Sep. 1975. doi: 10.1109/PROC.1975.9939

[2] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," ACM Transactions on Computer Systems, vol. 2, no. 4, pp. 277-288, 1984. doi: 10.1145/357401.357402

[3] U.S. Department of Housing and Urban Development, "Electronic Appraisal Delivery Portal: Hard Stop Checks and Error Messages (UAD 2.6)," HUD Office of Single Family Housing, revised Apr. 29, 2026; hard stop list current as of Dec. 10, 2024. [Online]. Available: https://www.hud.gov/sites/dfiles/SFH/documents/SFH_EAD_HARDSTOPSFS.pdf (accessed Sep. 28, 2026).

[4] U.S. Department of Housing and Urban Development, "FHA Uniform Appraisal Dataset MISMO Version 3.6 (UAD 3.6) Implementation Preparedness Toolbox," v1, HUD Office of Single Family Housing, Jun. 2026. [Online]. Available: https://www.hud.gov/sites/dfiles/SFH/documents/sfh_ead_FHA_UAD-3.6_Implementation_Preparedness_Toolbox_v1.pdf (accessed Sep. 28, 2026).

[5] R. Braden, Ed., "Requirements for Internet Hosts - Communication Layers," RFC 1122, RFC Editor, Oct. 1989. doi: 10.17487/RFC1122

[6] U.S. Department of Housing and Urban Development, "Modernizing FHA Appraisals: Transitioning to the New Uniform Appraisal Dataset 3.6 Standard," FHA INFO 2026-15, Jun. 25, 2026. FHA INFO archive, entry dated Jun. 25, 2026. [Online]. Available: https://www.hud.gov/hud-partners/single-family-fha-info (accessed Sep. 28, 2026).

[7] B. Meyer, "Applying 'design by contract'," Computer, vol. 25, no. 10, pp. 40-51, 1992. doi: 10.1109/2.161279

[8] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley, 2003. ISBN 978-0-321-20068-6.
[9] R. Yavatkar, D. Pendarakis, and R. Guerin, "A Framework for Policy-based Admission Control," RFC 2753, RFC Editor, Jan. 2000. doi: 10.17487/RFC2753
[10] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for Policy-Based Management," RFC 3198, RFC Editor, Nov. 2001. doi: 10.17487/RFC3198
[11] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu, "Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC)," in Proc. 2016 ACM International Workshop on Attribute Based Access Control, New Orleans, LA, USA, 2016, pp. 13-24. doi: 10.1145/2875491.2875496
[12] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis," in Proc. IEEE Annual Computer Software and Applications Conference (COMPSAC), 2012, pp. 233-243. doi: 10.1109/COMPSAC.2012.34
[13] J. Lehvä, N. Mäkitalo, and T. Mikkonen, "Consumer-driven contract tests for microservices: a case study," in Product-Focused Software Process Improvement, Lecture Notes in Computer Science, Cham, Switzerland: Springer, 2019, pp. 497-512. doi: 10.1007/978-3-030-35333-9_35
[14] S. Sadiq, G. Governatori, and K. Namiri, "Modeling Control Objectives for Business Process Compliance," in Business Process Management, Lecture Notes in Computer Science, Berlin, Germany: Springer, 2007, pp. 149-164. doi: 10.1007/978-3-540-75183-0_12
[15] S. Sadiq and G. Governatori, "Managing regulatory compliance in business processes," in Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture, Berlin, Germany: Springer, 2010, pp. 159-175. doi: 10.1007/978-3-642-01982-1_8
[16] M. Hashmi, G. Governatori, H.-P. Lam, and M. T. Wynn, "Are we done with business process compliance: state of the art and challenges ahead," Knowledge and Information Systems, vol. 57, no. 1, pp. 79-133, 2018. doi: 10.1007/s10115-017-1142-1
[17] T. Butler and L. O'Brien, "Understanding RegTech for digital regulatory compliance," in Disrupting Finance: FinTech and Strategy in the 21st Century, Cham, Switzerland: Palgrave Macmillan, 2018, pp. 85-102. doi: 10.1007/978-3-030-02330-0_6
[18] E. Micheler and A. Whaley, "Regulatory technology: replacing law with computer code," European Business Organization Law Review, vol. 21, no. 2, pp. 349-377, Jun. 2020. doi: 10.1007/s40804-019-00151-1
[19] B. Mihaylov, L. Onea, and K. M. Hansen, "Architecture-based regulatory compliance argumentation," Journal of Systems and Software, vol. 119, pp. 1-30, 2016. doi: 10.1016/j.jss.2016.04.057
[20] O. Kosenkov, P. Elahidoost, T. Gorschek, J. Fischbach, D. Mendez, M. Unterkalmsteiner, D. Fucci, and R. Mohanani, "Systematic mapping study on requirements engineering for regulatory compliance of software systems," Information and Software Technology, vol. 178, art. no. 107622, Feb. 2025. doi: 10.1016/j.infsof.2024.107622