HMPI
HEALTH MANAGEMENT, POLICY & INNOVATION

Appendix: Technical Requirements for a Computable Payer–Provider Contract & Digital Adjudication Platform

Lauren Bilbo, MBA (1), Elman Amador Medina (1), Dashiell Miner (2), David Scheinker, Ph.D. (2), Stefanos Zenios, PhD (1), Kevin Schulman, MD (1,2)

(1) Graduate School of Business, Stanford University

(2) School of Medicine, Stanford University
January, 2026

Support: The Ludy Family Foundation, The Hirsch Family Foundation, The Mindshare Institute, and the Government, Business and Society of the Graduate School of Business, Stanford University.

1. Abstract

Purpose

Define the technical, data, security, and governance requirements for implementing a computable, modular payer–provider contract and an accompanying digital transaction / adjudication platform.

Audience

Health system executives, provider-sponsored plan leaders, payer contract & claims operations teams, standards bodies (HL7, X12), regulators, health tech entrepreneurs, and investors evaluating infrastructure plays.

Scope

Focus on commercial & provider-sponsored plan medical claims (professional + facility) and core transaction classes: eligibility, coverage determination, network qualification, prior authorization (PA), claim submission & adjudication, remittance/payment, appeals, and population/outcomes-based value components. Pharmacy, dental, and behavioral carve-outs are out of initial scope except insofar as structural design must accommodate them in later modules.

Non-Goals

We do **not** propose a new clinical terminology (reuse LOINC/SNOMED/CPT/ICD), nor a wholesale replacement of X12/FHIR; instead we specify a *contract layer* that binds legal, financial, and operational semantics to executable logic referencing existing standards.

2. Problem Context & Motivation

In previous papers, we have discussed the possibility of creating modularized machine readable contracts to simplify the adjudication process [1,2]. These contracts would be modifiable between different payers and providers. By using Standard-Setting Organizations (SSOs), competing firms can come together to collectively innovate this modular machine readable contract into an industry standard. Alternatively, a public or public benefit corporation could design a uniform payment infrastructure to be used across the industry. These solutions would aid in creating a standard in the adjudication process and simplifying the existing administrative complexity in the healthcare industry.

To establish our technical framework, we audited a representative contract between a large health system partner (referred to here as “HealthSystem”) and a health plan (referred to as “InsuranceCompany”), isolating the specific clauses and variables necessary for a computable transaction platform. This analysis

allowed us to isolate both the standardized data elements common to the industry and the unique variables required for complex health system operations.

By synthesizing these findings with broader industry benchmarks, we have identified the foundational parameters necessary for a digital transaction form to achieve deterministic adjudication. This paper presents our research results, outlining a framework designed to bridge the gap between static legal prose and executable machine logic.

3. Standards & Data Layer

3.1 The Role of HL7 FHIR in Digital Contracting

To achieve seamless interoperability across the healthcare continuum from patient to provider to payer, the digital contracting platform must utilize a modular and standardized data architecture. HL7 Fast Healthcare Interoperability Resources (FHIR) R5 provides the necessary framework for representing clinical, administrative, and infrastructure entities. By integrating FHIR resources natively, the platform ensures that data remains liquid and actionable across different stakeholders without the need for extensive translation layers.

3.2 Transitioning from Legacy X12 to FHIR Native

While X12 remains the current industry standard for administrative data exchange (e.g., claims and eligibility), the regulatory landscape is shifting toward a unified FHIR-based ecosystem.

As mandated by the CMS Interoperability and Prior Authorization Final Rule, payers will be required to modernize to the FHIR standard by Jan 1, 2027 [3]. To do this, organizations will need to do 3 things.

1. Have a computable form that suggests which services are covered by any given plan and whether prior authorization is required for this service.
2. For each covered service that requires prior authorization, to create a form that includes all necessary information to complete authorization
3. Allow providers to submit the required documentation through a FHIR related system

To bridge the gap between current state and future requirements, the proposed platform is designed as FHIR-native. This ensures long-term scalability while maintaining backward compatibility with X12 to support legacy systems during the industry-wide transition period.

4. Concept Overview

A Modular Computable Contract (MCC) is a machine-readable bundle consisting of:

1. **Core Metadata Module:** Includes parties, term, jurisdictions, legal boilerplate, local law overlays, version IDs, and cryptographic hashes.
2. **Eligibility & Coverage Module**: Translates contract definitions into three interconnected validation engines (Service Coverage, Member Eligibility, and Benefit Precedence) to verify if services qualify for payment.
3. **Network & Facility Module**: Enforces participation requirements through automated TIN/NPI validation and includes a "Rate Parity Monitor" to automatically adjust payments if a payer contracts with non-qualified facilities.
4. **Prior Authorization Module**: Includes PA triggers, required clinical data elements, decision logic (CQL/FHIRPath rulesets), and the "immunity layer" for approved authorizations and SLAs.
5. **Payment Processes Module**: Automates financial fulfillment via an executable pipeline that handles intake, "Clean Claim" validation, and the automated calculation of interest for late payments or discount forfeitures.
6. **Fee Schedule & Rates Module**: Reconciles multiple pricing mechanisms, such as Percent-of-Billed-Charge, fixed schedules, and conditional modifiers, into a coherent data model for deterministic base rate lookups
7. **Contract Lifecycle Management Module**: Serves as the administrative control plane, version control via signed commits, and an automated reasoning validator to detect logical conflicts.
8. **Denials Module**: Implements a multi-layered framework that disables specific denial codes (e.g., lack of notification) and provides a stateful reprocessing queue for providers pending credentialing.
9. **Appeals & Adjustments Module**: Orchestrates digital workflows for dispute resolution, enforces contractual adjustment windows, and appends all events to an immutable event ledger for forensic traceability.

Each module is independently versioned and referenced via stable URIs so payers/providers can negotiate only the changed primitives (e.g., update PA triggers) without re-parsing entire documents.

5. Technical Requirements by Functional Area

5.1 Eligibility & Coverage Module

The Eligibility & Coverage module translates contract definitions into executable logic that validates whether services qualify for payment. This module implements three interconnected validation engines that mirror the contract's coverage specifications.

Service Coverage Validation Engine

Contract Clause: "Covered Services means professional and facility services that are covered under a Health Benefit Program. The parties understand and agree that services that are integral to the effective delivery of care, are medically necessary, and are being rendered in an appropriate setting, will be Covered Services and payable under this Agreement."

Implementation: The module implements a hierarchical decision tree where each service undergoes three sequential validations:

```
Input: CPT/HCPCS Code, ICD-10, Facility Type
→ Benefit Catalog Lookup (FHIR PlanDefinition)
→ Medical Necessity Rules Engine (CQL expressions)
→ Setting Appropriateness Validator
→ Output: Coverage Decision with Reason Codes
```

The challenge lies in the phrase "integral to the effective delivery of care", which requires contextual analysis beyond simple code matching. The module addresses this by implementing a dependency graph where certain services (anesthesia, infusion, pharmacy) are automatically covered when linked to a primary covered procedure. For example, when CPT 99213 (office visit) includes J3420 (vitamin B injection), the injection is covered as integral to care delivery even if not explicitly listed in benefits.

Member Eligibility Verification Engine

Contract Clause: "If InsuranceCompany incorrectly verifies the eligibility of a Member, InsuranceCompany or a Purchaser under contract with InsuranceCompany will pay the HealthSystem Facility or Professional Provider as though the Member was fully eligible for Covered Services rendered through the date that HealthSystem is notified of the error."

Implementation: This liability provision requires an audit-capable eligibility system:

```
Eligibility Request (X12 270) → Verification Engine → Response (X12 271)
↓
Immutable Audit Log
- Timestamp of verification
- Member ID and coverage details
- Cryptographic proof of response
```

The system maintains versioned eligibility snapshots to handle retroactive changes. When an incorrect verification is discovered, the module automatically flags all affected claims for payment under the liability provision, bypassing standard denial logic.

Health Benefit Program Precedence

Contract Clause: "To the extent inconsistency or conflict exists between the terms in InsuranceCompany's Health Benefit Program and this Agreement, the terms in this Agreement prevail."

This creates a two-tier validation system where the module must apply contract-specific overrides to the benefit definitions from the Health Benefit Program.

Proposed Solution: Implement a precedence engine using a top-down priority list:

```
Rule Priority Stack:
1. Agreement-specific overrides (highest priority)
2. Health Benefit Program defaults
3. Standard coverage policies (lowest priority)
```

If a conflict is detected at Layer 2, the engine automatically suppresses the default value in favor of the Layer 1 override. The system generates a Governance Log for every decision, documenting exactly which layer provided the governing rule to ensure auditability.

5.2 Network & Facility Module

The Network & Facility module enforces complex participation requirements through multiple validation layers, including automated breach detection and payment adjustments.

Network Composition Validator

Contract Clause: "The terms of this Agreement apply only to Covered Services rendered by a Professional Provider at an HealthSystem Facility and billed for by HealthSystem using the HealthSystem Tax Identification Number (TIN) 94-2854057."

Implementation: The module implements compound validation logic:

```
function validateNetworkService(claim):
    provider = lookupProvider(claim.renderingNPI)
    facility = lookupFacility(claim.serviceLocation)
    billingEntity = validateTIN(claim.billingTIN)

    isProviderValid = provider is in ProfessionalProviderList
    isFacilityValid = facility is in HealthSystemFacilities
    isBillingValid = billingEntity.TIN == "94-2854057"

    if isProviderValid and isFacilityValid and isBillingValid:
        return true
```

```
    else:
        return false
```

Automated Rate Parity & Adjustment

Contract Clause: "If InsuranceCompany makes agreements with non-HealthSystem facility healthcare providers beyond the terms of Section II.A, the payment rates in Attachment B will be automatically adjusted (reduced), effective as of the non-HealthSystem contract's effective date."

Implementation: This requires continuous monitoring of InsuranceCompany's contracting activities:

```
RateParityMonitor {
  scanNewContracts() → detectNonQualifiedFacilityContracts()
  ↓
  IF detected:
    - Calculate rate reduction percentage
    - Update all affected fee schedules
    - Timestamp effective date
    - Generate compliance alert
}
```

Complex Challenge: Detecting contracts with non-approved facilities requires external data feeds or self-reporting mechanisms. The module implements:

1. Integration with state insurance department filings
2. Provider directory change monitoring
3. Claims pattern analysis to detect new facility usage

Professional Provider List Management

Contract Clause: "HealthSystem will provide quarterly updates to the Professional Provider List, which are incorporated into the Agreement without amendment."

Implementation: The module maintains a versioned provider registry:

```
ProviderRegistry {
  Q1_2024_List: [providers...]
  Q2_2024_List: [providers + changes...]

  getProviderStatus(NPI, serviceDate) {
    list = getListVersionForDate(serviceDate)
    return list.contains(NPI)
  }
}
```

Non-Obvious Challenge: The contract states services by "non-employed, non-contracted professional providers are not subject to this Agreement," but doesn't specify how to identify these providers. The module addresses this by:

1. Flagging any NPI not on the Professional Provider List
2. Creating a secondary validation queue for manual review
3. Maintaining an exclusion list updated through provider credentialing

5.3 Prior Authorization Module

The Prior Authorization module manages complex authorization workflows while implementing absolute protection for approved services.

Service-Specific Authorization Requirements

Contract Clause: "HealthSystem is not required to obtain pre-authorization or pre-notification for outpatient hospital services (this includes outpatient observation), or for any services rendered by HealthSystem Homecare and Hospice."

Implementation: The module maintains authorization requirement matrices:

```
AuthorizationMatrix {
  "OUTPATIENT_HOSPITAL": NO_AUTH_REQUIRED,
  "OUTPATIENT_OBSERVATION": NO_AUTH_REQUIRED,
  "HOMECARE_*": NO_AUTH_REQUIRED,
  "INPATIENT_NON_URGENT": PRE_AUTH_REQUIRED,
  "INPATIENT_EMERGENT": POST_NOTIFICATION_REQUIRED
}
```

Authorization Protection Framework

Contract Clause: "Once HealthSystem obtains a pre-authorization, pre-certification, authorization, certification, a reference number, including a 278 EDI transaction response or any other number that could be interpreted as a pre-certification, pre-authorization, authorization, certification or reference number, InsuranceCompany agrees it will not reconsider the Medical Necessity of such service."

Implementation: This broad protection requires capturing all possible authorization formats:

```
AuthorizationCapture {
  sources: [
    "278 Response",
    "Portal Reference Number",
    "Voice Auth Code",
    "Email Confirmation",
```

```
    "System-Generated ID"
  ]

  protectService(auth) {
    // Create immutable record
    authRecord = {
      id: auth.referenceNumber,
      type: auth.source,
      timestamp: now(),
      hash: sha256(auth)
    }

    // Link to claim with override flag
    claim.authProtected = true
    claim.bypassMedicalNecessity = true
  }
}
```

Complex Challenge: The phrase "any other number that could be interpreted as authorization" creates ambiguity. The module addresses this by:

1. Implementing pattern recognition for authorization-like strings
2. Maintaining a whitelist of known authorization formats
3. Flagging ambiguous cases for human review
4. Defaulting to protection when uncertainty exists

Clinical Review Workflow Engine

Contract Clause: "InsuranceCompany agrees to request only the HealthSystem clinical utilization review report... Within one (1) working day of receiving the report, InsuranceCompany must communicate an appropriate number of certified days."

Implementation: The module enforces strict timelines through workflow orchestration:

```
ClinicalReviewWorkflow {
  receiveUtilizationReport(report) {
    startTimer(24_HOURS)

    if (!responseReceived && timerExpired) {
      autoCertify(DEFAULT_DAYS)
      logComplianceViolation()
    }
  }
}
```

5.4 Payment Processes Module

This Payment Processes module automates the financial fulfillment of the contract by transforming static payment terms into an executable logic pipeline.

Payment Terms

The generic agreement specifies a payment processes with multiple timing and penalty clauses that must become executable rules:

Contract Clause	Computable Rule Element	Data Inputs	Outcome
Clean Claim paid within 30 days or interest accrues (Sec. V.A & V.B.1.a)	payment_due = claim_receipt + 30d → if now > payment_due then interest = unpaid_amount × 0.00033 × late_days	Claim header (receipt date), unpaid balance	Auto-calculate interest line item; append to 835/ClaimResponse
Discount forfeiture after 121 days (V.B.1.b)	If `late_days ≥ 121` then allowed = 100 % × billed_charges	Billed charges, late_days	Overrides all other pricing logic
Automatic 15-day extension upon written notice (V.B.1.a)	extension_granted flag + new payment_due = claim_receipt + 45d	Extension request timestamp	Prevents premature late-payment logic
Direct-payment enforcement & penalty uplift (V.C)	Detect non-direct payment → apply penalty_rate_adjustment (+X %) until compliance	Payment remittance source flag	Dynamic adjustment of RateFormula set
Annual auto-escalator for CMS-1500 rates (V.A.2.b)	rate = base_rate × (1 + escalator)^{years}	Service date, effective date	Auto-update fee schedule version at runtime

Refund/recoup window limits (12/24/36 mo) (V.D & V.P)	Permit adjustment_request only if within window; otherwise reject	Payment date, adjustment request date	Deterministic acceptance/rejection of recoupment

Payment Processing Pipeline

1. **Intake & Validation**: Accept UB-04/NUBC or CMS-1500 via X12 837; ensure claim qualifies as *Clean* (all mandatory data present).
2. **Pricing Engine**: Apply contracted RateFormula or percent-of-charge logic *unless overridden* by late-payment or penalty rules.
3. **Penalty Layer**: Evaluate timing, accuracy, and discount-forfeiture conditions; inject interest or rate overrides as separate AdjustmentComponent objects for full traceability.
4. **835 / ClaimResponse Generation**: Provide line-item breakdown: base allowance, penalties, interest, member cost-share. Include explanationCodes mapped to CARC /RARC for each adjustment.
5. **Posting & Funds Movement**: Support EFT CCD+ remittance; persist transaction hash linking to executed rule bundle for audit.

Edge-Case Automation

- *One-Reason-At-A-Time Denials*: System is configured to always emit **all** denial reasons in the first response to preclude serial denial loops (§ V.B).
- *New Service / Technology Pricing*: When service.code not found in fee schedule, default to percent-of-charge rate per Attachment B (§ V.J) while triggering RateGapAlert for contracting review.

5.5 Fee Schedule & Rates Module

The agreement employs three distinct pricing mechanisms that a digital transaction platform must model and reconcile:

1. **Percent-of-Billed-Charge (PoC) formulas** that vary by both place of service and facility category.
2. **Fixed fee schedules** for professional (physician/APP) services and select outpatient procedures.
3. **Conditional discount modifiers** that raise or lower the base rate when contractual conditions are breached (e.g., steering patients to non-designated facilities).

From a systems perspective, the challenge is not calculating any single formula; it is supporting them all in a coherent data model so the engine can pick the right one for each claim.

Data-Model Approach

All rate instructions are stored in one table (or FHIR RateFormula resource) with columns:

```
serviceSetting   // INPATIENT, OUTPATIENT, ASC, etc.

facilityType     // URBAN_HOSPITAL, RURAL_HOSPITAL, LAB, etc.

rateMethod       // FIXED, PERCENT_OF_CHARGE, PERCENT_OF_MEDICARE

fixedRateValue        // decimal (e.g., 350.00)

percentageRateValue        // decimal (e.g., 0.70)

cap, floor       // optional monetary limits

version, dates   // for historical replay
```

A second table stores ConditionalAdjustment rows triggered by network violations, late payment, quality bonuses, etc. The adjudication engine simply:

1. Looks up the best-match *RateFormula* row for the claim's setting and facility type.
2. Applies any conditional adjustments relevant to that claim context.

Simplified Alternative

The mixed-method model above is fully computable, but operationally heavy. A simpler path is to standardize on a single base rate method:

- **Fixed fee** (one dollar amount per CPT/DRG code), **or**
- **Percent of Medicare** (e.g., 150 % of the CMS rate).

With that restriction, pricing collapses to two deterministic steps:

1. **Base rate lookup** – retrieve the fee schedule line (fixed dollar or Medicare multiplier).
2. **Apply conditional discount modifiers** – add or subtract percentage points if contract conditions (network status, timely filing, quality metrics) are met.

This two-step model greatly eases validation, forecasting, and conflict resolution, and can still accommodate incentives via the modifier layer.

5.6 Contract Lifecycle Management Module

The Contract Lifecycle Management module serves as the administrative control plane of the platform, ensuring proper version control. The module ensures that every amendment is tracked, validated, and cryptographically secured.

Authoring & Version Control UI

The design includes a web-based editor designed to eliminate the translation gap between legal counsel and IT. This web editor includes the following capabilities.

Bimodal Editing: A web-based "Split View" editor displays Legal Markdown (human-readable text) on the left and the corresponding JSON-LD/FHIR Bundle (machine-readable logic) on the right. Changes in one pane are reflected in the other in real-time.

Modular Clause Library: A drag-and-drop sidebar contains standardized templates for common contractual elements (e.g., *Percent-of-Charge* or *Late-Payment Interest*). Authors can insert these modules and adjust parameters via simple form controls, which automatically update the underlying logic.

Version Control & Non-Repudiation: Every modification is committed to a Git-like repository, capturing the `commitID`, author, and a `diffSummary`. To ensure absolute integrity, these commits are signed and anchored in an append-only hash ledger, preventing unauthorized retroactive changes.

Branching & Conflict Validation: Stakeholders utilize feature branches to draft specific amendments. Prior to merging, an Automated Reasoning Validator scans the new rules against the existing contract to detect logical conflicts or mathematical overlaps.

Machine-Readable Tagging

To enable cross-platform interoperability, the contract is serialized into two high-fidelity formats:

FHIR Bundle (Document Type): The primary serialization uses FHIR R5 resources, with section entries referencing custom StructureDefinitions for specialized data like RateFormula and AuthorizationRule.

Semantic JSON-LD: A parallel view enables semantic querying via SPARQL or JSONPath. This layer links contract terms to global medical vocabularies (LOINC, CPT, ICD-10) using Compact URIs (CURIEs), ensuring the platform understands exactly which clinical codes trigger specific payment rules.

Integrity Check: Both representations carry an identical SHA-256 hash in their metadata. Any discrepancy between the human-readable document and the machine-readable logic immediately blocks the contract from moving to the production environment.

Lifecycle States & Governance

The contract progresses through a strictly defined state machine, governed by role-based access controls and digital signatures. Transitions require digital signatures from designated roles (Legal, Actuary, IT Ops). EffectiveDate drives rule-activation scheduler.

State	Definition	Required Action
DRAFT	Initial authoring phase	No logic active; sandbox testing only
REVIEW	Multi-stakeholder validation	Signatures from Legal, Actuarial, and IT
APPROVED	Finalized version	Logic queued for deployment
EFFECTIVE	Active production state	**Rule-Activation Scheduler** triggers logic on EffectiveDate
DEPRECATED	Superseded by a newer version	No new claims accepted; open claims finalized
ARCHIVED	Final record retention	Read-only storage for audit and litigation

5.7 Denials Module

The Denials module implements a multi-layered framework that prevents inappropriate denials while ensuring comprehensive reporting when denials are warranted. The module addresses retrospective denial prevention, credentialing-based payment workflows, and member protection mechanisms.

Retrospective Denial Prevention Engine

Contract Clause: "Once HealthSystem obtains a pre-authorization, pre-certification, authorization, certification, or a reference number, InsuranceCompany agrees it will not reconsider the Medical Necessity or eligibility of such service or treatment and deny payment."

Implementation: The module creates an authorization immunity layer that overrides all subsequent denial logic:

```
RetrospectiveDenialBlocker {
  evaluateClaim(claim) {
    // Check for any form of prior authorization
    if (hasAnyAuthorization(claim)) {
      return {
        decision: "MUST_PAY",
        overrideReasons: ["PRIOR_AUTH_PROTECTION"],
        bypassChecks: ["MEDICAL_NECESSITY", "ELIGIBILITY"]
      }
    }
    // Continue normal evaluation only if no auth exists
    return standardEvaluation(claim)
  }

  hasAnyAuthorization(claim) {
    return claim.has278Response() ||
           claim.hasReferenceNumber() ||
           claim.hasPreCertification() ||
           claim.hasAnyAuthIndicator()
  }
}
```

Notification Penalty Exemption

Contract Clause: "InsuranceCompany agrees to pay claims for Covered Services in accordance with the agreement terms, and no sanctions or penalties will be imposed for lack of notification by HealthSystem."

Implementation: The module disables all notification-related denial codes:

```
NotificationExemption {
  denialReasons.remove("NO_PRIOR_NOTIFICATION")
  denialReasons.remove("LATE_NOTIFICATION")
  denialReasons.remove("IMPROPER_NOTIFICATION")

  // Log for audit but do not deny
  if (notificationMissing) {
    auditLog.record("Notification not provided - payment required per contract")
  }
}
```

Credentialing-Based Denial and Reprocessing Workflow

Contract Clause: "Claims for Non-Delegated Professional Providers may initially be denied or processed as non-participating if InsuranceCompany's credentialing requirements are not satisfied. However, InsuranceCompany must ensure such claims are reprocessed and paid as participating once the Non-Delegated Professional Provider meets InsuranceCompany's credentialing requirements, not to exceed thirty (30) days."

Implementation: This requires a stateful reprocessing queue with automated triggers:

```
CredentialingWorkflow {
  initialClaimProcessing(claim) {
    provider = getProvider(claim.NPI)

    if (!provider.isCredentialed && !provider.isHospitalBased) {
      // Create provisional denial with reprocessing flag
      return {
        status: "PENDED_FOR_CREDENTIALING",
        reprocessDate: now() + 30_DAYS,
        originalClaimId: claim.id
      }
    }
  }

  onCredentialingComplete(provider) {
    // Auto-trigger reprocessing of all pended claims
    pendedClaims = getPendedClaimsForProvider(provider.NPI)
    pendedClaims.forEach(claim -> {
      reprocessAsParticipating(claim)
      notifyPayment(claim)
    })
  }

  // Failsafe: After 30 days, allow member billing
  credentialingTimeout() {
    if (daysSincePended > 30) {
      releaseToBillMember(claim)
    }
  }
}
```

Hospital-Based Provider Protection

Contract Clause: "Hospital-based healthcare professionals (e.g., emergency medicine, radiology, anesthesia, pathology) providing services incident to hospital services do not require further credentialing by InsuranceCompany, and claims incurred with such healthcare providers will in no case be denied for lack of participation."

Implementation: The module maintains a specialty-based exemption list:

```
HospitalBasedExemption {
  exemptSpecialties = [
    "EMERGENCY_MEDICINE",
    "RADIOLOGY",
    "ANESTHESIOLOGY",
    "PATHOLOGY"
  ]

  if (provider.specialty in exemptSpecialties &&
      claim.placeOfService == "HOSPITAL") {
    claim.bypassCredentialingCheck = true
    claim.forceParticipatingRate = true
  }
}
```

Member Balance Billing Protection with Exceptions

Contract Clause: "HealthSystem Facilities and Professional Providers generally agree not to balance bill Members for Covered Services, except for applicable copayment, coinsurance, and/or deductible amounts. However, this exception does not apply in the event of non-payment by InsuranceCompany or a Purchaser."

Implementation: The module tracks payment status and manages billing permissions:

```
BalanceBillingController {
  evaluateMemberBillingRights(claim) {
    if (claim.status == "UNPAID" && claim.ageInDays > 30) {
      // Enable member billing after collection efforts
      return {
        memberBillable: true,
        reason: "PAYER_NON_PAYMENT",
        requiredNotices: generateMemberNotices()
      }
    }
```

```
    if (eligibilityResponse.hasBalanceBillProtection) {
      return {
        memberBillable: false,
        reason: "LEGAL_PROTECTION"
      }
    }
  }

  generateMemberNotices() {
    return [
      "HealthSystem has billed InsuranceCompany",
      "Payment has not been received",
      "Member is financially responsible"
    ]
  }
}
```

Pediatric Imaging Special Protections

Contract Clause: "InsuranceCompany agrees that it will not apply any software edits (e.g., CCI, Claim Check, iCES) to claims that would bundle or unbundle procedures and result in the denial of services or reduction in payment of claims for IPI."

Implementation: The module creates provider-specific edit exemptions:

```
PediatricImagingProtection {
  if (claim.provider == "HEALTHSYSTEM_PEDIATRIC_IMAGING") {
    disableEdits([
      "CCI_BUNDLING",
      "CLAIM_CHECK",
      "iCES_EDITS"
    ])

    // Auto-approve medical necessity
    if (claim.hasOrderingPhysician) {
      claim.medicallyNecessary = true
      claim.authorized = true
    }
  }
}
```

Comprehensive Denial Response Framework

When denials are permitted, the module ensures all reasons are provided simultaneously:

```
ComprehensiveDenialEngine {
  generateDenialResponse(claim) {
    denialReasons = []

    // Run all checks unless blocked by protection rules
    if (!claim.hasAuthProtection) {
      denialReasons.addAll(checkAllCriteria(claim))
    }

    if (denialReasons.isEmpty()) {
      return approvePayment(claim)
    }

    return {
      status: "DENIED",
      reasons: denialReasons,
      carcCodes: mapToCARCRARC(denialReasons),
      appealRights: generateAppealInstructions(),
      adjustmentDeadline: now() + 12_MONTHS
    }
  }
}
```

Complex Implementation Challenges:

1. **"Inappropriate Denial" Adjustments:** The contract allows 12-month windows for adjustment requests but requires "mutual agreement." The module implements a dispute resolution queue where both parties can submit evidence, with escalation to human review when automated reconciliation fails.
2. **Funding Deficiency Handling:** When InsuranceCompany lacks funds, the module must switch from standard denial processing to immediate member billing rights, requiring integration with financial monitoring systems to detect insolvency indicators.
3. **Multi-Tiered Protection Rules:** The interplay between authorization protection, notification exemption, and credentialing workflows creates complex precedence rules that the module manages through a hierarchical decision engine where protective provisions always override denial logic.

5.8 Appeals & Adjustments Module

The contract gives each side limited windows to question a payment: 12 months for most errors, 24 months for coordination-of-benefits, 36 months when government claw-backs apply. It also bars retrospective medical-necessity denials on services that already received prior auth and routes unresolved disputes to binding arbitration.

These rules could be implemented as follows:

1. **Adjustment Service** - Validates against rule window; auto-rejects late requests.
2. **Denial Guardrail** - Prior-auth flag attaches to claim header; pricing engine blocks medical-necessity denials when flag is set.
3. **Dispute Module** - After two levels of internal appeal, unresolved items are flagged for arbitration.
4. **Event Ledger** - All adjustment, audit, and dispute events append to an immutable hash-chain linked to the original claim, ensuring forensic traceability.

6. Non-computable elements

The digital adjudication platform must acknowledge that certain contract provisions require human judgment, discretion, and negotiation that cannot be fully automated. These elements require hybrid human-machine workflows that maximize automation while preserving necessary human oversight.

Medical Necessity Determinations

Non-Computable Aspect: The contract defines Medical Necessity as healthcare services "a prudent physician would provide for preventing, diagnosing, or treating an illness... in accordance with generally accepted standards of medical practice." This clinical judgment cannot be fully automated.

Proposed Hybrid Solution:

```
MedicalNecessityWorkflow {
  // Automated pre-screening
  automatedTriage(claim) {
    if (meetsClearCriteria(claim)) {
      return AUTO_APPROVE  // routine cases
    }
    if (clearlyExcluded(claim)) {
      return AUTO_DENY     // obvious exclusions
    }
    return HUMAN_REVIEW    // require clinical judgment
  }
```

```
  // AI-assisted human review
  humanReviewInterface {
    - Present clinical guidelines
    - Show similar case precedents
    - Highlight relevant medical records
    - Require physician documentation of decision rationale
  }
}
```

Credentialing and Provider Verification

Non-Computable Aspect: Credentialing requires verification of medical licenses, board certifications, DEA licenses, hospital privileges, professional liability insurance, and malpractice history, all requiring human verification of external sources.

Proposed Hybrid Solution:

```
CredentialingAutomation {
  // Automated data collection
  gatherCredentials() {
    - API calls to state medical boards
    - Database queries to DEA registry
    - Electronic verification services
  }

  // Human verification required for:
  - Malpractice history interpretation
  - Hospital privilege confirmation calls
  - Final approval decision

  // Automated tracking
  trackExpirations() {
    - Alert 90 days before license expiry
    - Auto-queue for re-credentialing
  }
}
```

Appeals Process Management

Non-Computable Aspect: Appeals require human review of documentation, establishment of "mutually agreed upon written time frames," and negotiation between parties.

Proposed Hybrid Solution:

```
AppealsManagement {
  // Automated workflow orchestration
```

```
  appealWorkflow {
    - Electronic appeal submission portal
    - Automated acknowledgment and tracking
    - Document assembly and routing
    - Timeline monitoring and alerts
  }

  // Human requirements:
  - Clinical review of appeal merits
  - Negotiation of resolution timeframes
  - Final determination documentation

  // Automation assists:
  - Template generation for common scenarios
  - Precedent database for consistency
  - Automated implementation of decisions
}
```

Audit Execution and Oversight

Non-Computable Aspect: Physical audits require human auditors to review claims and medical records, with HealthSystem having "sole discretion" to approve or deny audit requests.

Proposed Hybrid Solution:

```
AuditFramework {
  // Pre-audit automation
  prepareAudit() {
    - Statistical sampling of claims
    - Automated outlier detection
    - Document compilation
    - Risk scoring for focus areas
  }

  // Human audit activities:
  - On-site record reviews
  - Provider interviews
  - Discretionary access decisions

  // Post-audit automation
  implementFindings() {
    - Automated claim adjustments
    - System rule updates
    - Tracking of corrective actions
  }
}
```

Purchaser Delinquency Assessment

Non-Computable Aspect: Identifying "consistent and unresolved issues" and determining if resolution timeframes are "reasonable" requires qualitative human judgment.

Proposed Hybrid Solution:

```
DelinquencyMonitoring {
  // Automated metrics
  trackPaymentPatterns() {
    - Days outstanding trends
    - Payment variance analysis
    - Communication response rates
  }

  // Human decision points:
  - Evaluate "reasonableness" of delays
  - Strategic termination decisions
  - Negotiation of payment plans

  // System support:
  - Dashboard of key indicators
  - Automated warnings at thresholds
  - Documentation of decisions
}
```

Administrative Performance Reviews

Non-Computable Aspect: Annual rate reviews based on "administrative performance" require subjective evaluation of both quantitative metrics and qualitative factors.

Proposed Hybrid Solution:

```
PerformanceReview {
  // Automated scorecard
  generateMetrics() {
    - Clean claim rate
    - Payment timeliness
    - Authorization turnaround
    - Appeal overturn rates
  }

  // Human evaluation of:
  - Communication quality
```

```
  - Problem resolution effectiveness
  - Strategic partnership value

  // Structured decision support:
  - Benchmark comparisons
  - Trend analysis
  - Rate adjustment calculators
}
```

Balance Billing and Collection Decisions

Non-Computable Aspect: Determining "reasonable commercial efforts" for collection and deciding when to bill members directly requires case-by-case human judgment.

Proposed Hybrid Solution:

```
CollectionDecisionSupport {
  // Automated tracking
  monitorCollectionEfforts() {
    - Number of attempts
    - Communication methods used
    - Payment promise tracking
  }

  // Human decisions:
  - Define "reasonable" for each case
  - Approve member billing
  - Handle special circumstances

  // System assists:
  - Template collection notices
  - Automated timeline documentation
  - Member communication tracking
}
```

Financial Assistance Waivers

Non-Computable Aspect: HealthSystem may waive copayments based on "financial standing," "willingness to expedite payment," or "otherwise", all requiring individualized human assessment.

Proposed Hybrid Solution:

```
FinancialAssistanceWorkflow {
  // Automated screening
  initialAssessment() {
    - Income verification via databases
```

```
    - Federal poverty level calculations
    - Payment history analysis
  }

  // Human determinations:
  - Evaluate financial hardship
  - Assess payment willingness
  - Apply discretionary waivers

  // Standardization support:
  - Decision matrices for common scenarios
  - Documentation templates
  - Approval workflow routing
}
```

Contract Amendments and Negotiations

Non-Computable Aspect: Amendments require negotiation, consensus-building, legal drafting, and physical signatures, inherently human processes.

Proposed Hybrid Solution:

```
AmendmentManagement {
  // Digital collaboration platform
  - Version-controlled amendment drafts
  - Comment and revision tracking
  - Automated impact analysis

  // Human requirements:
  - Negotiation of terms
  - Legal review and approval
  - Strategic decision-making

  // Post-agreement automation:
  - System rule updates
  - Affected claim identification
  - Retroactive adjustments
}
```

Implementation Recommendations

1. **Design for Human-in-the-Loop:** Build interfaces that present relevant information efficiently for human decision-makers while automating data gathering and implementation.
2. **Create Decision Audit Trails:** Document all human decisions with structured rationales to build precedent databases that can guide future automation.
3. **Implement Graduated Automation:** Start with clear-cut cases and gradually expand automated decision-making as patterns emerge from human decisions.
4. **Establish Override Mechanisms:** Ensure humans can always override automated decisions when circumstances warrant exception handling.
5. **Monitor for Automation Opportunities:** Continuously analyze human decisions to identify patterns that could be codified into rules for future automation.

The platform should embrace these non-computable elements as opportunities for human expertise to add value while maximizing automation of routine, rule-based processes. This hybrid approach ensures both efficiency and the flexibility needed for complex healthcare contracting.

7. Change Management & Policy

Change management reflects the ongoing process of implementing new practices, systems or policies within any given hospital, clinic or health system. In this case, implementing a digital contracting platform within the healthcare industry would require some industry standardization or intervention via public or public-benefit corporation.

Incentivizing insurance companies to standardize the adjudication process may be tricky. As previously stated, insurance companies often benefit from the complexity of the adjudication process. Using SSOs and Standard Essential Patents (SEPs) could incentivize an engagement in creating a digital contracting platform. SSOs are often composed of industry partnerships that bring together different companies to develop some kind of market standard for the rest of the industry. This could be developed by America's Health Insurance Plans (AHIP) or by the federal government. Within this SSO, insurance companies can compete to develop a standard for digital contracts and the best innovation is considered the SEP. The company that develops the SEP will receive royalties from other firms in the market.

An alternative to the SSO/SEP process would be to create a uniform payment structure. This could be performed by a public or public-benefit corporation. This system would set up a centralized payment infrastructure with accessibility though APIs. Such an infrastructure could hold the modular digital contracts and could enforce the stipulations throughout payment. The utilization of either of these methods would greatly simplify the adjudication process and cut out administrative waste in the healthcare industry.

Conclusion

This document establishes a technical and operational framework for transitioning from traditional, healthcare agreements to a computable, modular payer–provider contract system. By deconstructing contracts into discrete, machine-executable modules, such as fee schedules, network requirements, and prior authorization rules, this framework aims to eliminate the administrative "opacity" that currently drives high denial rates and systemic waste.

The core findings and recommendations are summarized as follows:

- **Modular Architecture:** Future contracts should be structured into nine functional modules: Core Metadata, Eligibility & Coverage, Network & Facility, Prior Authorization, Payment Processes, Fee Schedule & Rates, Contract Lifecycle Management, and Denials. This allows for the isolation of high-variance negotiated terms from stable operational logic.
- **Industry Standardization:** Adopting a standard, modular, computable contract can provide a shared foundation for machine-readable terms, enabling payers and providers to customize economic logic without re-engineering the entire adjudication engine.
- **Technological Integration**: Achieving "touchless" adjudication requires a robust rules-based engine, API-driven interoperability (utilizing FHIR and X12 standards), and real-time validation of patient eligibility and service documentation.
- **Operational Impact:** Moving toward a centralized payment infrastructure or an industry-wide Standard Essential Patent (SEP) model for digital contracts will streamline the adjudication process, reduce manual labor, and ultimately lower administrative costs across the healthcare ecosystem.

In conclusion, the shift to computable contracts represents a fundamental evolution in healthcare infrastructure, moving the industry toward a transparent, deterministic, and highly efficient digital adjudication environment.

References

1. Brooke Istvan, Perry Nielsen Jr, Megan Eluhu, Bryan Kozin, Walt Winslow, David Scheinker, Kavita Patel, Kenneth Favaro, Stefanos Zenios, Kevin Schulman. 2024. Applying Precedents Thinking to the Intractable Problem of Transaction Costs in Healthcare. Health Management, Policy and Innovation (www.HMPI.org). Volume 9, Issue 3.
2. Istvan B, Schulman KA, Zenios S. Addressing Health Care's Administrative Cost Crisis. JAMA. 2025 Mar 4;333(9):749-750.
3. https://www.cms.gov/newsroom/fact-sheets/cms-interoperability-and-prior-authorization-final-rule-cms-0057-f