**Regulatory Compliance IntelligenceTool Specification**

**Tool Purpose & Integration**

**The Complete Solution Stack**

AI Agent (with regulatory context)

↓ (queries regulatory knowledge)

RAG Knowledge Base (regulatory documents)

↓ (provides regulatory requirements)

AI Agent (understands applicable regulations)

↓ (calls tool with transaction\_id)

RegulatoryComplianceIntelligenceTool

↓ (fetches submission data)

MongoDB Submission Database

↓ (performs compliance analysis)

Structured Compliance Assessment

↓ (returns to agent)

AI Agent (synthesizes final regulatory intelligence report)

**Tool Class Definition (Following Your Blueprint)**

class RegulatoryComplianceIntelligenceTool(BaseTool):

"""

Tool for analyzing commercial insurance submissions against applicable

regulatory requirements to identify compliance gaps and generate

actionable recommendations.

"""

# Studio-required metadata

name = "RegulatoryComplianceIntelligenceTool"

description = "Analyzes commercial insurance submissions for regulatory compliance gaps across federal, state, and local regulations"

requires\_env\_vars = [

"MONGO\_CONNECTION\_STRING: mongodb+srv://artifi:root@artifi.2vi2m.mongodb.net/?retryWrites=true&w=majority&appName=Artifi"

]

dependencies = [("pandas", "pandas"), ("pymongo", "pymongo"), ("re", "re")]

uses\_llm = False

default\_llm\_model = None

default\_system\_instructions = None

structured\_output = True

# Schema definitions

input\_schema = {

"type": "object",

"properties": {

"transaction\_id": {

"type": "string",

"description": "The unique transaction ID (artifi\_id) to retrieve submission data from MongoDB"

}

},

"required": ["transaction\_id"]

}

output\_schema = {

"type": "object",

"properties": {

"compliance\_summary": {

"type": "object",

"properties": {

"overall\_compliance\_score": {"type": "number"},

"total\_regulations\_assessed": {"type": "integer"},

"critical\_gaps": {"type": "integer"},

"high\_priority\_gaps": {"type": "integer"},

"medium\_priority\_gaps": {"type": "integer"},

"regulatory\_risk\_level": {"type": "string"},

"estimated\_resolution\_timeline": {"type": "string"}

}

},

"jurisdictional\_analysis": {

"type": "object",

"description": "Compliance analysis by jurisdiction",

"additionalProperties": {

"type": "object",

"properties": {

"applicable\_regulations": {"type": "array"},

"compliance\_score": {"type": "number"},

"gaps\_identified": {"type": "array"},

"recommendations": {"type": "array"}

}

}

},

"compliance\_gaps": {

"type": "array",

"items": {

"type": "object",

"properties": {

"gap\_id": {"type": "string"},

"regulation\_domain": {"type": "string"},

"severity": {"type": "string"},

"description": {"type": "string"},

"business\_impact": {"type": "string"},

"remediation\_steps": {"type": "array"},

"timeline": {"type": "string"},

"estimated\_cost": {"type": "number"}

}

}

},

"regulatory\_opportunities": {

"type": "array",

"items": {

"type": "object",

"properties": {

"opportunity\_type": {"type": "string"},

"description": {"type": "string"},

"potential\_benefit": {"type": "string"},

"implementation\_effort": {"type": "string"}

}

}

},

"submission\_analysis": {

"type": "object",

"properties": {

"company\_name": {"type": "string"},

"locations\_analyzed": {"type": "array"},

"industries\_identified": {"type": "array"},

"construction\_types": {"type": "array"},

"occupancy\_types": {"type": "array"},

"special\_risk\_factors": {"type": "array"}

}

},

"transaction\_id": {"type": "string"},

"error": {"type": "string"}

},

"required": ["compliance\_summary", "jurisdictional\_analysis", "compliance\_gaps", "submission\_analysis", "transaction\_id"]

}

**Core Processing Logic**

**1. Submission Data Extraction**

def \_extract\_regulatory\_context(self, submission\_data):

"""Extract regulatory-relevant information from submission"""

context = {

"locations": self.\_extract\_all\_locations(submission\_data),

"industries": self.\_extract\_naics\_codes(submission\_data),

"occupancies": self.\_extract\_occupancy\_types(submission\_data),

"construction\_types": self.\_extract\_construction\_classes(submission\_data),

"building\_characteristics": self.\_extract\_building\_features(submission\_data),

"coverage\_types": self.\_extract\_coverage\_details(submission\_data),

"operational\_details": self.\_extract\_business\_operations(submission\_data),

"company\_info": self.\_extract\_company\_details(submission\_data)

}

return context

**2. Regulatory Requirements Mapping**

def \_assess\_federal\_compliance(self, context):

"""Assess compliance with federal regulations"""

federal\_assessment = {

"flood\_insurance": self.\_check\_fema\_requirements(context),

"accessibility": self.\_check\_ada\_requirements(context),

"workplace\_safety": self.\_check\_osha\_requirements(context),

"environmental": self.\_check\_epa\_requirements(context)

}

return federal\_assessment

def \_assess\_state\_compliance(self, context):

"""Assess compliance with state-specific regulations"""

state\_assessments = {}

for location in context["locations"]:

state = location.get("state")

if state:

state\_assessments[state] = {

"building\_codes": self.\_check\_state\_building\_codes(location, context),

"insurance\_regulations": self.\_check\_state\_insurance\_reqs(location, context),

"catastrophe\_requirements": self.\_check\_catastrophe\_reqs(location, context)

}

return state\_assessments

**3. Compliance Gap Analysis**

def \_identify\_compliance\_gaps(self, assessment\_results, context):

"""Identify specific compliance gaps with risk scoring"""

gaps = []

# Federal compliance gaps

for domain, assessment in assessment\_results["federal"].items():

if assessment.get("compliant") is False:

gap = {

"gap\_id": f"FED\_{domain.upper()}\_{len(gaps)+1}",

"regulation\_domain": f"Federal {domain.title()}",

"severity": assessment.get("severity", "medium"),

"description": assessment.get("gap\_description"),

"business\_impact": assessment.get("business\_impact"),

"remediation\_steps": assessment.get("remediation\_steps", []),

"timeline": assessment.get("timeline", "90 days"),

"estimated\_cost": assessment.get("cost\_estimate", 0)

}

gaps.append(gap)

# State compliance gaps

for state, domains in assessment\_results["state"].items():

for domain, assessment in domains.items():

if assessment.get("compliant") is False:

gap = {

"gap\_id": f"{state}\_{domain.upper()}\_{len(gaps)+1}",

"regulation\_domain": f"{state} {domain.title()}",

"severity": assessment.get("severity", "medium"),

"description": assessment.get("gap\_description"),

"business\_impact": assessment.get("business\_impact"),

"remediation\_steps": assessment.get("remediation\_steps", []),

"timeline": assessment.get("timeline", "90 days"),

"estimated\_cost": assessment.get("cost\_estimate", 0)

}

gaps.append(gap)

return sorted(gaps, key=lambda x: {"critical": 0, "high": 1, "medium": 2, "low": 3}[x["severity"]])

**4. Specific Compliance Checks**

**FEMA Flood Requirements**

def \_check\_fema\_requirements(self, context):

"""Check FEMA flood insurance compliance"""

assessment = {"compliant": True, "gaps": []}

for location in context["locations"]:

# Check if location is in flood zone

if self.\_is\_flood\_zone(location):

building\_value = location.get("building\_value", 0)

if building\_value > 500000: # Threshold from synthetic data

if not location.get("flood\_insurance"):

assessment["compliant"] = False

assessment["severity"] = "critical"

assessment["gap\_description"] = f"Property in flood zone lacks required flood insurance (value: ${building\_value:,})"

assessment["business\_impact"] = "Coverage denial risk, lender requirement violation"

assessment["remediation\_steps"] = [

"Obtain flood zone determination letter",

"Purchase NFIP flood insurance",

"Verify elevation certificate requirements"

]

assessment["timeline"] = "30 days"

assessment["cost\_estimate"] = 5000

return assessment

**Texas Windstorm Requirements**

def \_check\_texas\_windstorm\_requirements(self, location, context):

"""Check Texas windstorm insurance compliance"""

assessment = {"compliant": True}

if location.get("state") == "TX" and self.\_is\_coastal\_texas(location):

# Check for windstorm mitigation features

required\_features = ["impact\_resistant\_windows", "reinforced\_roof", "storm\_shutters"]

missing\_features = []

for feature in required\_features:

if not location.get(feature):

missing\_features.append(feature)

if missing\_features:

assessment["compliant"] = False

assessment["severity"] = "high"

assessment["gap\_description"] = f"Missing required windstorm mitigation: {', '.join(missing\_features)}"

assessment["business\_impact"] = "50% rate penalty, potential coverage denial"

assessment["remediation\_steps"] = [

"Schedule wind mitigation inspection",

"Install missing mitigation features",

"Obtain certified compliance documentation"

]

assessment["timeline"] = "120 days"

assessment["cost\_estimate"] = 25000

return assessment

**ADA Accessibility Requirements**

def \_check\_ada\_requirements(self, context):

"""Check ADA accessibility compliance"""

assessment = {"compliant": True}

for location in context["locations"]:

building\_size = location.get("square\_footage", 0)

if building\_size > 5000: # Commercial building threshold

# Check required accessibility features

if not location.get("accessible\_parking"):

assessment["compliant"] = False

assessment["severity"] = "high"

assessment["gap\_description"] = "Missing required accessible parking spaces"

assessment["business\_impact"] = "DOJ enforcement risk, potential $75,000+ penalties"

assessment["remediation\_steps"] = [

"Conduct accessibility audit",

"Install required accessible parking",

"Ensure compliant signage and markings"

]

assessment["timeline"] = "60 days"

assessment["cost\_estimate"] = 15000

return assessment

**Business Logic Features**

**Risk Scoring Algorithm**

def \_calculate\_compliance\_score(self, gaps):

"""Calculate overall compliance score based on gaps"""

if not gaps:

return 100

penalty\_weights = {

"critical": 25,

"high": 15,

"medium": 8,

"low": 3

}

total\_penalty = sum(penalty\_weights.get(gap["severity"], 0) for gap in gaps)

return max(0, 100 - total\_penalty)

**Timeline Estimation**

def \_estimate\_resolution\_timeline(self, gaps):

"""Estimate overall timeline to resolve all gaps"""

if not gaps:

return "Currently compliant"

critical\_gaps = [g for g in gaps if g["severity"] == "critical"]

high\_gaps = [g for g in gaps if g["severity"] == "high"]

if critical\_gaps:

return "30-60 days (critical issues require immediate attention)"

elif high\_gaps:

return "3-6 months (high priority compliance gaps)"

else:

return "6-12 months (standard compliance improvements)"

**Opportunity Identification**

def \_identify\_regulatory\_opportunities(self, context, gaps):

"""Identify opportunities for premium discounts and competitive advantages"""

opportunities = []

# Look for potential discount opportunities

for location in context["locations"]:

if location.get("state") == "TX" and location.get("sprinkler\_system"):

opportunities.append({

"opportunity\_type": "premium\_discount",

"description": "Enhanced fire protection qualifies for 15% rate reduction",

"potential\_benefit": "15% premium savings (~$5,000-$15,000 annually)",

"implementation\_effort": "Documentation review only"

})

if location.get("seismic\_retrofit") and location.get("state") == "CA":

opportunities.append({

"opportunity\_type": "premium\_discount",

"description": "Seismic retrofit completion qualifies for 20% rate reduction",

"potential\_benefit": "20% premium savings (~$8,000-$25,000 annually)",

"implementation\_effort": "Engineering certification required"

})

return opportunities

**Integration with Existing Patterns**

**Following PropertyValuationTool Pattern**

* **Same input pattern**: transaction\_id only
* **Same MongoDB access**: Uses identical connection and query patterns
* **Same output structure**: Structured JSON response with error handling
* **Same blueprint compliance**: All required metadata and schemas

**Complementary to PropertyValuationTool**

* **PropertyValuationTool**: Analyzes "Is the property value correct?"
* **RegulatoryComplianceIntelligenceTool**: Analyzes "Does the property meet regulatory requirements?"
* **Together**: Complete risk assessment for underwriting decisions

**Expected Tool Output Example**

{

"compliance\_summary": {

"overall\_compliance\_score": 72,

"total\_regulations\_assessed": 23,

"critical\_gaps": 1,

"high\_priority\_gaps": 2,

"medium\_priority\_gaps": 3,

"regulatory\_risk\_level": "High",

"estimated\_resolution\_timeline": "3-6 months"

},

"jurisdictional\_analysis": {

"Federal": {

"applicable\_regulations": ["FEMA Flood", "ADA Accessibility", "OSHA Safety"],

"compliance\_score": 85,

"gaps\_identified": ["Missing flood insurance documentation"],

"recommendations": ["Obtain flood zone determination", "Purchase NFIP coverage"]

},

"TX": {

"applicable\_regulations": ["Windstorm Insurance", "Building Code"],

"compliance\_score": 60,

"gaps\_identified": ["Missing windstorm mitigation", "Inadequate roof attachment"],

"recommendations": ["Install impact windows", "Upgrade roof system"]

}

},

"compliance\_gaps": [

{

"gap\_id": "TX\_WINDSTORM\_1",

"regulation\_domain": "Texas Windstorm Insurance",

"severity": "critical",

"description": "Property lacks required windstorm mitigation features",

"business\_impact": "50% rate penalty, potential coverage denial",

"remediation\_steps": ["Schedule wind mitigation inspection", "Install impact windows"],

"timeline": "60 days",

"estimated\_cost": 25000

}

],

"regulatory\_opportunities": [

{

"opportunity\_type": "premium\_discount",

"description": "Sprinkler system qualifies for fire protection discount",

"potential\_benefit": "15% premium reduction",

"implementation\_effort": "Documentation only"

}

],

"submission\_analysis": {

"company\_name": "ABC Manufacturing Inc",

"locations\_analyzed": ["Houston, TX", "Los Angeles, CA"],

"industries\_identified": ["Manufacturing - NAICS 332"],

"construction\_types": ["Noncombustible"],

"occupancy\_types": ["Manufacturing", "Warehouse"],

"special\_risk\_factors": ["Coastal location", "Seismic zone"]

},

"transaction\_id": "ABC123"

}

**Complete Solution Workflow**

**How All Components Work Together**

1. **Agent receives request**: "Analyze regulatory compliance for transaction ABC123"
2. **Agent queries RAG knowledge base**:
   * "Texas windstorm requirements manufacturing"
   * "FEMA flood insurance commercial properties"
   * "ADA accessibility requirements"
3. **Agent calls RegulatoryComplianceIntelligenceTool**:
   * RegulatoryComplianceIntelligenceTool(transaction\_id="ABC123")
4. **Tool performs structured analysis**:
   * Fetches submission from MongoDB
   * Extracts regulatory context
   * Performs compliance checks
   * Returns structured assessment
5. **Agent synthesizes final report**:
   * Combines RAG regulatory context with tool analysis
   * Generates comprehensive regulatory intelligence report
   * Provides business-focused recommendations

This tool completes the regulatory intelligence story by providing the analytical bridge between your knowledge base and practical compliance assessment.