OMRC Execution-Quality Surveillance: POC Implementation Approach

Executive Summary

This document outlines a Proof of Concept (POC) strategy to implement the OMRC (Off-Market/Off-Rate Check) framework as described in the comprehensive SOP. The POC demonstrates execution-quality surveillance capabilities through a phased, product-agnostic approach that validates contextual signal collection, intent inference, and explainable alert resolution.

POC Objectives:

- Validate context-driven alert explanation logic
- Demonstrate probabilistic intent classification
- Prove economic plausibility assessment mechanisms
- Establish analyst workflow integration
- Generate audit-defensible audit trails

---

1. POC Scope & Constraints

In Scope

- Single product line pilot (recommend: FX Spot + FX Forward)
- Alert trigger mechanization (deviation detection)
- Core context signal collection (5-7 signals per product)
- Intent likelihood scoring (non-binary classification)
- Economic plausibility rules engine
- Analyst decision recording interface

Out of Scope

- Product lineage assertion (explicitly excluded per SOP)
- Hedge validation automation
- Front-office system replacement
- Auto-closure mechanisms
- KMRC or other surveillance frameworks

Duration & Team

- **Phase 1 (Weeks 1-2):** Data architecture + signal design
- **Phase 2 (Weeks 3-5):** Model development + rule engine
- **Phase 3 (Weeks 6-7):** Testing + UAT
- **Team Size:** 2-3 engineers (data, Python), 1 analyst (domain), 1 architect

---

2. POC Architecture Overview

2.1 High-Level Flow

Trade Execution (Exchange/System)
↓
[Alert Trigger] — Market Deviation Detection
↓
[Signal Collection] — Context Aggregation (ETL)
↓
[Intent Inference] — Probabilistic Classification (ML)
↓
[Economic Plausibility] — Rules-Based Assessment
↓
[Alert Ranking] — Risk Scoring
↓
[Analyst Review] — Manual Decision + Rationale Recording
↓
[Audit Trail] — Decision Justification

2.2 Technology Stack

Data Layer:

- **Source:** CSV/Parquet export from trading systems (Refinitiv, Bloomberg, internal OMS)
- **Storage:** Python pandas DataFrames (POC phase) → PostgreSQL (production)
- **ETL:** Python scripts with Streamlit UI for data loading

Processing Layer:

- **Core Logic:** Python (scikit-learn for signal normalization, scoring)
- **Rules Engine:** Custom Python class-based implementation
- **Intent Model:** Regression-based probability scoring (no heavy ML needed for POC)

Interface Layer:

- **Streamlit** dashboard for analyst workflow (familiar to your tech stack)
- **Data validation:** Regex-based checks for trade data integrity
- **Audit logging:** SQLite database for decision trails

---

3. POC Phase 1: Data Architecture & Signal Design

3.1 Alert Trigger Mechanization

Trade data schema (minimum required):

Field	Type	Source	Purpose
trade_id	String	OMS	Unique identifier
trade_date	DateTime	OMS	Execution timestamp
product	String	OMS	FX Spot / FX Forward / IRS
executed_rate	Float	OMS	Execution price/rate
market_mid	Float	Market data	Real-time or end-of-day mid
principal	Float	OMS	Trade notional/size
currency_pair	String	OMS	EUR/USD, GBP/JPY, etc.
counterparty_type	String	OMS	Client / Hedge / Interbank

Deviation detection (baseline threshold):

FX Spot/Forward: |executed_rate - market_mid| > N bps (threshold: 3-5 bps)
IRD (IRS): |executed_rate - curve_fair_value| > N bps (threshold: 10-15 bps)
Bonds: |executed_yield - benchmark_yield| > N bps (threshold: 15-25 bps)

Recommended thresholds (calibrate in UAT):

- Tier 1 Alert: > 5 bps (FX) — High severity
- Tier 2 Alert: 3-5 bps (FX) — Standard review
- Tier 3 Alert: 1-3 bps (FX) — Low severity (informational)

3.2 Core Context Signals (Per Product)

FX Spot/Forward Signals:

Signal	Calculation	Rationale
size_percentile	(principal / 30-day avg daily volume) × 100	Large size justifies deviation
liquidity_hour	Is trade in illiquid window? (Asia close, NY open)	Low liquidity explains slippage
same_book_exposure	Sum of same-currency positions in desk book	Hedge execution pattern
historical_hedge_ratio	Count(back-to-back trades with same currency) / total trades (30 days)	Client facilitation likelihood
time_to_settlement	Days between trade and settlement date	Funding management signal
client_facilitation_flag	Is counterparty a retail/institutional client?	Context for execution justification
market_volatility	Realized volatility 5-min window around trade	High vol. justifies execution discretion

IRD Signals (IRS Example):

Signal	Calculation	Rationale
curve_deviation	(executed_rate - interpolated_curve_mid) / curve_std_dev	Normalized deviation magnitude
back_to_back_detection	Same notional + tenor + opposite direction within 5 sec?	Mechanical execution not speculative
tenor_concentration	Count(same tenor in 1 hour)	Concentrated activity pattern
spread_vs_history	Z-score of bid-ask spread vs 30-day avg	Deviation from normal conditions
funding_corridor_flag	Is trade in stressed funding window (EOD/MOY/EOQ)?	Funding-driven execution
counterparty_hedge_history	Historical hedge ratio with this counterparty	Established pattern signal

Bonds Signals (Cash Bonds Example):

Signal	Calculation	Rationale
yield_deviation_zscore	(executed_yield - benchmark_yield) / yield_std_dev	Statistical significance
illiquidity_premium	Estimated liquidity cost vs benchmark	Off-the-run premium justification
position_roll_pattern	Is bond near maturity? Related trades in past 30 days?	Inventory management signal
block_trade_size	Is principal > typical daily flow × 50%?	Block trade context

Canonical Signal Normalization (Python):

```
class SignalNormalizer:
def __init__(self, signal_config):
self.config = signal_config
```

```
def normalize_percentile(self, value, historical_values):
    """Rank value within historical distribution"""
    return np.percentileofscore(historical_values, value)

def normalize_zscore(self, value, mean, std):
    """Standardize to normal distribution"""
    return (value - mean) / std if std > 0 else 0

def normalize_binary(self, condition):
    """Boolean to 0/1"""
    return 1.0 if condition else 0.0
```

---

4. POC Phase 2: Intent Inference & Economic Plausibility

4.1 Intent Classification Framework

OMRC recognizes 5 non-binary intent categories with probabilistic scoring:

Total Probability = 100% (distributed across categories)

Intent Category	Definition	Key Signals	Probability Range
Hedge Execution	Offsetting desk risk from previous trades	same_book_exposure, historical_hedge_ratio, back_to_back_detection	0-100%
Client Facilitation	Accommodating client order requirements	client_facilitation_flag, counterparty_type, size_percentile	0-100%
Funding/Liquidity Management	Managing short-term cash/funding needs	funding_corridor_flag, time_to_settlement, spread_vs_history	0-100%
Portfolio/Inventory Management	Rebalancing desk inventory or duration	position_roll_pattern, tenor_concentration, position_aging	0-100%
Speculative/Risk-Taking	Directional market bet with no hedging	absence of hedge signals, concentrated positioning, historical pattern break	0-100%

4.2 Intent Likelihood Scoring Model

Probabilistic model (soft constraints, not deterministic):

class IntentInferenceEngine:
"""
Aggregates normalized signals to produce intent probabilities.
Uses logistic regression coefficients calibrated on historical analyst decisions.
"""

```
def __init__(self, model_coefficients):
    self.coefs = model_coefficients  # Fitted weights

def score_hedge_execution(self, signals):
    """
    High probability if:
```

```
    - Same-book exposure > 70th percentile
    - Back-to-back detection = True
    - Same counterparty history > 0.5
    """
    score = (
        self.coefs['same_book_exposure'] * signals['same_book_exposure'] +
        self.coefs['back_to_back'] * signals['back_to_back_flag'] +
        self.coefs['hedge_history'] * signals['counterparty_hedge_ratio']
    )
    return self._sigmoid(score)

def score_client_facilitation(self, signals):
    """
    High probability if:
    - Counterparty is retail/institutional client
    - Size correlates with typical client order magnitudes
    - Trade aligns with client book direction
    """
    score = (
        self.coefs['client_flag'] * signals['client_facilitation_flag'] +
        self.coefs['size_match'] * signals['size_vs_client_avg'] +
        self.coefs['book_alignment'] * signals['book_alignment_score']
    )
    return self._sigmoid(score)

def score_funding_management(self, signals):
    """
    High probability if:
    - Trade in stressed funding window (EOD/MOY/EOQ)
    - Time-to-settlement < 7 days (short-term funding need)
    - Spread anomaly vs historical norm
    """
    score = (
        self.coefs['funding_window'] * signals['funding_corridor_flag'] +
        self.coefs['settlement_timing'] * (1 - signals['time_to_settlement'] / 365) +
        self.coefs['spread_anomaly'] * signals['spread_zscore']
    )
    return self._sigmoid(score)
```

```
def _sigmoid(self, x):
    """Convert logit to probability"""
    return 1 / (1 + np.exp(-x))

def infer_intent_distribution(self, signals):
    """Normalize probabilities to sum to 1"""
    raw_scores = {
        'hedge': self.score_hedge_execution(signals),
        'client': self.score_client_facilitation(signals),
        'funding': self.score_funding_management(signals),
        'portfolio': self.score_portfolio_management(signals),
        'speculative': self.score_speculative(signals)
    }
    total = sum(raw_scores.values())
    return {k: v/total for k, v in raw_scores.items()}
```

4.3 Economic Plausibility Rules Engine

Explainable decision tree (prioritized rule evaluation):

class EconomicPlausibilityEngine:
"""
Applies hierarchical rules to assess whether execution deviation is economically defensible.
Rules are product-agnostic and order-independent (no deterministic assertion).
"""

```
def __init__(self):
    self.rules = []

def add_rule(self, name, condition_func, severity_reduction, rationale):
    """
    Add a plausibility rule.
    Severity reduction: How much does this rule mitigate alert severity (0-100%)?
    """
    self.rules.append({
        'name': name,
        'condition': condition_func,
        'severity_reduction': severity_reduction,
        'rationale': rationale
```

```
    })

def evaluate(self, trade_context, intent_distribution):
    """
    Determine plausibility score and explanation.
    """
    matched_rules = []
    total_severity_reduction = 0

    for rule in self.rules:
        if rule['condition'](trade_context, intent_distribution):
            matched_rules.append(rule)
            total_severity_reduction += rule['severity_reduction']

    plausibility_score = 100 - min(total_severity_reduction, 100)

    return {
        'score': plausibility_score,  # 0-100 (higher = more defensible)
        'matched_rules': [r['name'] for r in matched_rules],
        'rationales': [r['rationale'] for r in matched_rules],
        'final_decision': 'EXPLAINABLE' if plausibility_score > 60 else 'ESCALATE'
    }
```

POC Rule Examples

engine = EconomicPlausibilityEngine()

Rule 1: Large size in illiquid window

```
engine.add_rule(
name='Large_Size_Illiquid_Window',
condition_func=lambda ctx, intent: (
ctx['size_percentile'] > 75 and
ctx['liquidity_hour'] == True and
intent['hedge'] > 0.3
),
severity_reduction=70,
rationale='Large size (75th percentile) executed in illiquid window (Asia close) with hedge
context justifies 7 bps deviation'
)
```

Rule 2: Back-to-back neutralization (IRD)

```
engine.add_rule(
name='Back_to_Back_Neutralization',
condition_func=lambda ctx, intent: (
ctx.get('back_to_back_flag', False) == True and
abs(intent['hedge']) > 0.6
),
severity_reduction=85,
rationale='Back-to-back neutralization (same tenor/notional/opposite direction within 5 sec) is mechanical execution'
)
```

Rule 3: Off-the-run bond premium

```
engine.add_rule(
name='Illiquidity_Premium_Justification',
condition_func=lambda ctx, intent: (
ctx.get('illiquidity_premium', 0) > 10 and # > 10 bps premium
ctx['block_trade_size'] == True
),
severity_reduction=65,
rationale='Off-the-run bond yields reflect liquidity cost; block trade size warrants premium execution'
)
```

Rule 4: Funding corridor (end of day, month, quarter)

```
engine.add_rule(
name='Funding_Stress_Window',
condition_func=lambda ctx, intent: (
ctx.get('funding_corridor_flag', False) == True and
ctx.get('time_to_settlement', 365) < 7 and
intent['funding'] > 0.4
),
severity_reduction=60,
rationale='End-of-day funding pressure with short settlement justifies wider FX swap pricing'
)
```

4.4 Alert Risk Ranking

Composite scoring for analyst triage:

```
class AlertRiskRanker:
def score_alert(self, deviation_magnitude, plausibility_score, intent_distribution):
"""
```

Rank alert by residual risk after context evaluation.
Scale: 0-100 (higher = riskier)
"""

```
        # Base risk: normalized deviation magnitude
        base_risk = min(abs(deviation_magnitude) / 10 * 100, 100)

        # Plausibility discount: reduce risk if economically defensible
        plausibility_discount = plausibility_score

        # Intent risk adjustment
        speculative_weight = intent_distribution.get('speculative', 0) * 100
        hedge_weight = intent_distribution.get('hedge', 0) * (-50)  # Hedge lowers risk
        client_weight = intent_distribution.get('client', 0) * (-30)

        residual_risk = base_risk - plausibility_discount + speculative_weight + hedge

        return max(0, min(residual_risk, 100))

    def classify_severity(self, risk_score):
        """Map risk to operational tiers"""
        if risk_score > 70:
            return 'CRITICAL - Immediate escalation'
        elif risk_score > 50:
            return 'HIGH - Standard analyst review'
        elif risk_score > 30:
            return 'MEDIUM - Grouped review'
        else:
            return 'LOW - Informational, no action'
```

---

5. POC Phase 3: Analyst Workflow & Audit Trail

5.1 Streamlit-Based Review Interface

Analyst workflow (5-step SOP):

```
import streamlit as st
import pandas as pd
from datetime import datetime
```

```
class OMRCAnalystDashboard:
def __init__(self, alert_db):
self.alerts = alert_db
self.decisions = []
```

```
    def step_1_mechanical_trigger(self, alert_id):
        """Display raw alert: deviation magnitude, trigger reason"""
        st.subheader("Step 1: Mechanical Trigger Review")
        alert = self.alerts.get(alert_id)
        st.metric("Deviation (bps)", alert['deviation_bps'])
        st.metric("Alert Reason", alert['alert_reason'])
        st.write(f"Trade ID: {alert['trade_id']}")
        st.write(f"Executed Rate: {alert['executed_rate']}, Market Mid: {alert['market_r

    def step_2_liquidity_size(self, alert_id):
        """Assess size and liquidity context"""
        st.subheader("Step 2: Liquidity & Size Assessment")
        alert = self.alerts.get(alert_id)
        col1, col2, col3 = st.columns(3)
        with col1:
            st.metric("Principal (M)", alert['principal_millions'])
        with col2:
            st.metric("Size Percentile", f"{alert['size_percentile']:.1f}%")
        with col3:
            st.metric("Liquidity Hour?", "Yes" if alert['liquidity_hour'] else "No")

        st.write("Interpretation: Trades in the 90th percentile during Asia close are e

    def step_3_contextual_signals(self, alert_id):
        """Display AI-inferred context"""
        st.subheader("Step 3: Contextual Signals (AI-Enriched)")
        alert = self.alerts.get(alert_id)

        # Intent probabilities
        st.write("**Inferred Trade Intent (Probabilistic):**")
        intent_df = pd.DataFrame({
            'Intent': list(alert['intent_distribution'].keys()),
            'Probability': list(alert['intent_distribution'].values())
        })
```

```
        st.bar_chart(intent_df.set_index('Intent'))

        # Matched plausibility rules
        st.write("**Economic Plausibility Context:**")
        for rule in alert['matched_plausibility_rules']:
            st.write(f"✓ {rule}")

    def step_4_economic_plausibility(self, alert_id):
        """Analyst evaluates defensibility"""
        st.subheader("Step 4: Economic Plausibility Assessment")
        alert = self.alerts.get(alert_id)

        st.write(f"**Plausibility Score: {alert['plausibility_score']}/100**")
        st.write(f"**Model Recommendation: {alert['model_decision']}**")

        # Analyst override
        analyst_decision = st.radio(
            "Analyst Decision:",
            ["EXPLAINABLE", "ESCALATE", "REQUIRE_EVIDENCE"]
        )

        return analyst_decision

    def step_5_record_decision(self, alert_id, decision, analyst_id):
        """Audit trail: record rationale"""
        st.subheader("Step 5: Record Decision & Rationale")

        analyst_rationale = st.text_area(
            "Rationale (required):",
            placeholder="Explain your decision with specific references to signals, cor
        )

        supporting_evidence = st.file_uploader(
            "Attach supporting evidence (optional)",
            type=['pdf', 'xlsx', 'txt']
        )

        if st.button("Submit Decision"):
```

```
        decision_record = {
            'alert_id': alert_id,
            'analyst_id': analyst_id,
            'decision': decision,
            'rationale': analyst_rationale,
            'timestamp': datetime.now(),
            'evidence_file': supporting_evidence.name if supporting_evidence else N
        }
        self.decisions.append(decision_record)
        st.success("Decision recorded successfully")
        return decision_record

    return None
```

5.2 Audit Trail Schema

Decision database (audit-defensible):

```
class AuditTrailDB:
def init(self, db_path='omrc_audit.db'):
self.db = db_path
```

```
def schema(self):
    """
    SQLite schema for audit trail.
    Every decision is immutable and timestamped.
    """
    return """
    CREATE TABLE IF NOT EXISTS omrc_decisions (
        decision_id TEXT PRIMARY KEY,
        alert_id TEXT NOT NULL,
        trade_id TEXT NOT NULL,
        analyst_id TEXT NOT NULL,
        analyst_name TEXT,
        decision TEXT CHECK(decision IN ('EXPLAINABLE', 'ESCALATE', 'REQUIRE_
        analyst_rationale TEXT NOT NULL,
        model_recommendation TEXT,
        plausibility_score REAL,
        inferred_intent_json TEXT,
```

```
        matched_rules_json TEXT,
        supporting_evidence_path TEXT,
        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
        decision_justification_code TEXT,
        regulatory_reference TEXT,
        escalation_reason TEXT
    );

    CREATE TABLE IF NOT EXISTS omrc_model_audit (
        audit_id TEXT PRIMARY KEY,
        alert_id TEXT NOT NULL,
        model_version TEXT,
        signal_values_json TEXT,
        intent_probabilities_json TEXT,
        plausibility_rules_evaluated_json TEXT,
        severity_score REAL,
        model_decision TEXT,
        computation_timestamp DATETIME,
        signal_freshness_minutes INT
    );
    """

def record_decision(self, decision_data):
    """
    Atomic write of decision with full context.
    Supports regulatory audit: "show me how this alert was decided"
    """
    # SQL INSERT statement (pseudocode)
    # Ensures analyst rationale is mandatory for escalations
    pass

def audit_query(self, alert_id):
    """
    Full decision trace: alert → model signals → analyst decision → rationale
    """
    return {
        'alert_id': alert_id,
        'model_computation': self.get_model_audit(alert_id),
```

```
        'analyst_decision': self.get_decision(alert_id),
        'regulatory_alignment': self._map_to_sop_principle(alert_id)
    }
```

---

6. POC Testing & Validation Strategy

6.1 Test Data Design

Synthetic test scenarios (7-10 representative cases):

Scenario	Product	Alert Type	Expected Decision	Validation Metric
Large hedge during Asia close	FX Spot	6 bps deviation	EXPLAINABLE	Intent: hedge > 70%, plausibility: 75%
Back-to-back IRS neutralization	IRS	12 bps rate dev.	EXPLAINABLE	Back-to-back flag: True, intent: hedge 90%
Off-the-run bond premium	Bond	20 bps yield dev.	EXPLAINABLE	Illiquidity premium recognized, block trade
EOD FX swap funding stress	FX Swap	8 bps dev.	EXPLAINABLE	Funding window: True, settlement < 7 days
Concentrated spec positioning	IRS	25 bps dev.	ESCALATE	Intent: speculative 85%, no hedge hedge signal
Client facilitation mismatch	FX Spot	4 bps dev.	REQUIRE_EVIDENCE	Size < client avg, unclear justification
Outlier mark deviation	Bond	35 bps dev.	ESCALATE	Deviation > 2 std dev, no contextual signal

6.2 Validation Checklist

- [ ] **Signal Collection:** All 5-7 signals compute correctly for each product
- [ ] **Intent Scoring:** Probabilities sum to 100%, match domain expert judgment on 80%+ of cases
- [ ] **Plausibility Rules:** "True positives" (defensible) correctly identified; "false positives" (concerning) flagged for escalation
- [ ] **Analyst Workflow:** All 5 SOP steps can be completed within 3-5 min per alert
- [ ] **Audit Trail:** Decision rationale captured; reproducible on query
- [ ] **Model Governance:** Approved wording matches SOP section 13
- [ ] **Regulatory Alignment:** No auto-closure; no deterministic linkage assertions; human oversight preserved

6.3 UAT Sign-Off Criteria

- Middle office analysts validate signal interpretations (no "black box" concerns)
- Compliance confirms audit trail meets regulatory expectations
- Model risk team approves governance documentation
- Live trading desk endorses workflow (min 95% acceptance rate in pilot run)

---

7. POC Deliverables & Success Criteria

7.1 Core Deliverables

1. **Data Pipeline** (Python + Streamlit)
    - Trade data ingestion (CSV/OMS export)
    - Real-time market data enrichment (Bloomberg/Refinitiv API stubs)
    - Signal computation module (7 FX signals, 5 IRD signals, 5 Bond signals)
2. **Intent Inference Engine**
    - Logistic regression model (fitted on historical analyst decisions)
    - Output: intent probability distribution (non-deterministic)
3. **Economic Plausibility Rules Engine**
    - Hierarchical rule evaluator
    - Output: plausibility score (0-100) + matched rules + rationales
4. **Analyst Dashboard (Streamlit)**
    - 5-step SOP workflow interface
    - Alert triage and ranking
    - Decision input + rationale capture
5. **Audit Trail Database**
    - SQLite schema (decision_id, analyst_id, rationale, timestamp, evidence_path)
    - Query interface for regulatory traceability
6. **Documentation**
    - Technical SOP (signal definitions, rules, model governance)
    - Analyst training guide
    - Regulatory positioning memo (aligned with model governance section)

7.2 Success Criteria

Metric	Target	Measurement
Signal Coverage	100% (all 5-7 signals per product working)	Validation script: signal_audit_check()
Intent Accuracy	> 80% agreement with domain expert labeling on test set	Confusion matrix: predicted vs expert
Plausibility Decision Time	< 3 min per alert (Step 1-5 end-to-end)	Streamlit performance profiler
Analyst Acceptance	> 90% of pilot alerts reviewed without "unclear" feedback	Survey + UAT feedback log
Audit Trail Completeness	100% of escalations have rationale + timestamp	Audit query: missing_rationale_check()
Regulatory Compliance	Zero model governance violations	Compliance review checklist

---

8. Implementation Roadmap (7 Weeks)

Week 1-2: Data Architecture

- Define trade data schema + signal catalog
- Build trade data loader (CSV/OMS export)
- Implement signal normalization module
- Validate 3-5 signals on sample trades

Week 3-4: Model Development

- Fit intent inference model (logistic regression) on 500+ historical analyst decisions
- Develop economic plausibility rules engine (8-10 core rules)
- Build alert risk ranker

Week 5-6: Interface & Testing

- Develop Streamlit analyst dashboard (5-step workflow)
- Build SQLite audit trail database
- Conduct synthetic scenario testing (7-10 test cases)

Week 7: UAT & Documentation

- Middle office UAT (pilot run on 100 live/recent alerts)
- Compliance review + regulatory positioning
- Generate analyst training materials

---

9. Risk Mitigation & Contingencies

Risk	Impact	Mitigation
Market data latency	Alert signals outdated	Cache 5-min snapshot; accept 15-min delay in POC
Limited historical analyst decisions	Intent model overfits	Use domain expert rules as fallback; test on holdout set
Stakeholder misalignment	"Not what we expected"	Weekly demos; confirm 5-step SOP workflow in week 2
Signal calculation bugs	Unreliable output	Peer review all signal code; unit test each signal function
Audit trail performance	Large dataset slows queries	SQLite indexing on alert_id + timestamp; archive old records

---

10. Production Readiness (Post-POC)

If POC succeeds, production transition includes:

1. **Scale signal catalog** to all products (GFX, IRD, Bonds, Equities, Money Market)
2. **Migrate storage** from SQLite → PostgreSQL with replicated audit trail
3. **Integrate with front-office systems** (OMS, EMS, risk platform APIs)
4. **Regulatory reporting** (export OMRC decisions for supervisory review)
5. **Model governance framework** (retraining schedule, monitoring, explainability reviews)

---

11. Alignment with OMRC SOP Principles

This POC directly implements the OMRC SOP requirements:

SOP Principle	POC Implementation
Execution-quality focused	Alert trigger based on market deviation (rate/price), not product lineage
Context- and behavior-driven	Intent inference aggregates 5-7 contextual signals per product
Product-agnostic logic	Same 5-step SOP applies across FX, IRD, Bonds, Equities
No deterministic linkage	Intent probabilities are non-binary; no "parent transaction" assertion
Analyst in the loop	5-step dashboard enforces human review; no auto-closure
Explainable decisions	All plausibility rules are named, auditable, and rationale-documented
AI/ML support only	Model aggregates signals; analyst makes final decision
Audit defensible	Every decision timestamped, rationale recorded, evidence path tracked

---

12. Next Steps & Kickoff

1. **Week 1 Kickoff Meeting:**
   - Confirm pilot product (FX Spot + Forward recommended)
   - Assign data steward (trade data + market data access)
   - Establish UAT timeline
2. **Data Preparation:**
   - Extract 2-3 weeks of historical trades (500-1000 samples)
   - Gather market data snapshots for same period
   - Identify 5-10 analyst decisions for model training
3. **Stakeholder Alignment:**
   - Compliance review of audit trail schema
   - Trading desk endorsement of 5-step workflow
   - Middle office sign-off on signal definitions

References

- OMRC Comprehensive Alert Explanation SOP (All Products) — Source Document
- Section 4: Universal OMRC Alert Flow
- Section 12-13: AI/ML Role & Model Governance Positioning
- Section 14: Operational SOP for OMRC Analysts