creditX Complete Agent Catalogue v1.0

Master Registry for Production Build

Last Updated: January 15, 2026
Status: Production-Ready
Total Agents: 20 + Cross-cutting = 22 agents
Build Phase: Post-Agent 1, Pre-Agent 2 Detailed Specification

Table of Contents

Overview & Architecture

Agent Ecosystem Design

The creditX platform operates on a **multi-engine, role-aware agent orchestration model** where:

- **20 domain agents** operate within specific engines (Outcome, Rights & Trust, Risk & Security, Market & Capital)
- **2 cross-cutting agents** (Explainer, Notification) serve all faces and engines
- **3 face-specific views** (Consumer OS, Partner OS, Internal OS) determine which agents are visible/executable
- **Agent types** are classified as: **assistant** (read-only

recommendations), **operator** (internal state mutations), or **ambassador** (external actions)

Core Principles

1. **Policy-as-Code First** - Every agent decision is enforced by OPA/Cerbos policies
2. **Semantic APIs Only** - Agents query only pre-approved semantic entities, no free-form SQL
3. **Risk-Tiered HITL** - Human-in-loop gates scale from low-risk (automated) to high-risk (manual review required)
4. **Auditability** - Every agent action is logged with correlation IDs, decision reasons, and override tracking
5. **Outcome-Centric** - Agent success is measured by Net Advantage impact, fairness metrics, and SLA compliance

Agent Schema & Metadata

Standard Agent Registry Record

```
text
agent_id: "engine.agent_name.v{version}"
name: "Human-Readable Agent Name"
engine: "[outcome | rights_trust | risk_security | market_capital |
cross]"
agent_type: "[assistant | operator | ambassador]"
faces: "[consumer | partner | internal]" # One or more
scope: "Concise description of agent purpose"

# Input/Output Contracts
input_entities:
  - SemanticEntity1
  - SemanticEntity2
output_entities:
  - SemanticEntity3
  - SemanticEntity4

# Tool Access
required_tools:
  - semantic_query_tool_1
  - semantic_query_tool_2
  - external_api_tool_1

# Risk & Governance
risk_level: "[low | medium | high]"
requires_human_review: [true | false]
```

```
max_auto_actions_per_day: 100 # null for unlimited
max_auto_actions_per_workflow: 1 # per case/campaign

# Lifecycle
status: "[experimental | active | deprecated]"
milestone: "[M0–M5]" # Readiness level
created_at: "YYYY-MM-DD"
owner_team: "TeamName"

# Performance Tracking
avg_latency_ms: 500
success_rate_pct: 98.5
human_override_rate_pct: 2.1
monthly_invocations: 15000
```

ENGINE 1: OUTCOME ENGINE

Purpose: Turn financial goals into executable campaigns with measurable outcomes
Core Responsibility: Goal setting, plan generation, outcome tracking, campaign optimization
Agent Count: 4 agents
Materialized Views: mv_consumer_outcomes, mv_partner_outcomes, mv_campaign_performance, mv_org_outcome_summary

Agent 1: Plan Generation Agent

```
text
agent_id: outcome.plan_generation.v1
name: Plan Generation Agent
agent_type: ambassador
risk_level: high
faces: [consumer]
milestone: M3
status: active

scope: |
  Create personalized financial plans with measurable campaigns
  based on consumer goals, constraints, and current state

input_entities:
  - ConsumerGoals (active goals with priorities)
  - ConsumerConstraint (financial limits)
  - ConsumerSnapshot (current financial state)
  - ConsumerSecuritySummary (risk adjustments)

output_entities:
  - SavingsPlan (comprehensive plan)
  - ActionItems (step-by-step actions)
```

```
  - CampaignPlaybook (campaign structure)
  - PredictionOutputs (probability estimates)

required_tools:
  - get_consumer_snapshot
  - get_consumer_goals
  - get_constraints
  - generate_plan
  - calculate_net_advantage
  - estimate_probabilities

business_logic:
  Phase 1: Analysis
    1. Fetch consumer goals (primary + secondary)
    2. Prioritize goals based on impact & feasibility
    3. Get current financial state (income, debts, assets)
    4. Identify constraints (hard limits, preferences)

  Phase 2: Planning
    1. Calculate Net Advantage for each goal combo
    2. Generate campaign phases (90-day, 6-month, annual)
    3. Define action items with effort & impact
    4. Determine automation opportunities

  Phase 3: Probability & Risk
    1. Estimate success probability per action
    2. Calculate risk-adjusted Net Advantage
    3. Adjust for security posture (no high-risk plans if compromised)
    4. Incorporate behavioral insights

  Phase 4: Output
    1. Generate SavingsPlan with milestones
    2. Create campaign playbooks (Score Lift, Debt-Free, Tax-Smart)
    3. Estimate probabilities (95% confidence band)
    4. Queue notification for consumer

error_handling:
  - Missing data → Request consumer sync
  - Conflicting goals → Present ranking options
  - Impossible constraints → Flag and suggest relaxation
  - Probability < 30% → Warn consumer, require explicit acceptance

guardrails:
  - Max 3 active campaigns per consumer
  - Auto-executable actions require explicit opt-in
  - High-effort items (>40 hours) require review
  - Debt-related plans require debt validation

requires_human_review: false (auto-generated for consumer review)
max_auto_actions: unlimited (recommendations only)

example_output:
  plan_id: plan_18249384
  consumer_id: cons_29384
  net_advantage: 4500 # $4500 projected improvement
```

```
  confidence: 0.78 # 78% confidence band
  active_campaigns:
    - campaign_type: "Score Lift Track"
      target: 40_point_increase
      horizon: 90_days
      key_actions: ["pay_off_card", "dispute_inaccuracies",
"update_profile"]
```

Agent 2: Outcome Evaluation Agent

```
text
agent_id: outcome.evaluation.v1
name: Outcome Evaluation Agent
agent_type: operator
risk_level: low
faces: [internal]
milestone: M3
status: active

scope: |
  Track actual vs. projected outcomes for plans and campaigns,
  update tracking records, and trigger notifications

input_entities:
  - SavingsPlan (original plan)
  - OutcomeTracks (historical tracking)
  - ConsumerSnapshot (current state)
  - CampaignPlaybook (active campaigns)

output_entities:
  - OutcomeTracks (updated tracking)
  - PredictionOutputs (recalibrated estimates)
  - MilestoneEvents (achieved milestones)
  - CampaignPerformanceMetrics (aggregate stats)

required_tools:
  - get_outcome_tracks
  - update_outcome_tracks
  - calculate_net_advantage
  - detect_milestones
  - enqueue_notification

business_logic:
  Daily/Weekly Execution (low-latency):
    1. Fetch all OutcomeTracks with refresh_due=true
    2. Get latest ConsumerSnapshots
    3. Compare projected vs. actual progress
    4. Update OutcomeTracks (% completion, velocity)
    5. Detect milestones (25%, 50%, 75%, 100%)

  Periodic Recalibration (monthly):
    1. Recalculate Net Advantage with new data
    2. Adjust projections based on actual velocity
    3. Flag stalling campaigns (< 10% progress/month)
    4. Recommend campaign pivots
```

```
  Outcome Prediction (real-time on query):
    1. Retrieve PredictionOutputs for this consumer
    2. Compare against actuals
    3. Adjust confidence bands
    4. Return updated predictions

error_handling:
  - Stale consumer data → Request sync
  - Missing milestones → Check data integrity
  - Negative progress → Flag for support review
  - Campaign stalled > 60 days → Auto-pause

guardrails:
  - Never change plan without consumer approval
  - Milestone notifications require user opt-in
  - Performance metrics are read-only until recalibration approved

requires_human_review: false
max_auto_actions: unlimited

example_output:
  track_id: track_39284
  consumer_id: cons_29384
  plan_id: plan_18249384
  metric: "Score Lift"
  baseline: 680
  current: 698
  target: 720
  progress_pct: 46.7
  velocity_points_per_week: 2.1
  milestones_achieved: [25, 50]
  recalibrated_confidence: 0.82
```

Agent 3: Campaign Tuning Agent

```
text
agent_id: outcome.campaign_tuning.v1
name: Campaign Tuning Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M4
status: active

scope: |
  Analyze campaign performance and recommend config changes
  to playbooks (e.g., adjust action sequencing, timings)

input_entities:
  - CampaignOutcomeSummary (aggregate performance)
  - CampaignPlaybook (current playbook config)
  - PredictionOutputs (prediction accuracy per segment)
  - ConsumerSegmentStats (cohort performance)
```

```
output_entities:
  - CampaignConfigRecommendations (proposed changes)
  - RecalibrationReport (before/after metrics)
  - ExperimentProposals (A/B test suggestions)

required_tools:
  - get_campaign_summary
  - get_prediction_outputs
  - simulate_config_change
  - draft_config_changes
  - propose_experiment

business_logic:
  Analysis (weekly):
    1. Aggregate campaign outcomes by segment
    2. Calculate completion rates, velocity, outcomes
    3. Compare against baseline playbook
    4. Identify underperforming actions/sequences

  Optimization (monthly):
    1. Run simulations on proposed changes (adjust delays, costs,
sequencing)
    2. Estimate impact (projected outcome improvement)
    3. Model sensitivity (which changes matter most)
    4. Rank by expected ROI

  Experimentation:
    1. Propose champion/challenger playbooks
    2. Calculate required sample size for significance
    3. Generate experiment proposal for approval

error_handling:
  - Insufficient data → Require 30+ cohort members
  - Unstable metrics → Flag and skip optimization
  - Simulation failure → Use baseline estimates

guardrails:
  - Never deploy campaign changes without approval
  - A/B tests require statistical power calculation
  - Changes must preserve fairness metrics

requires_human_review: true (all recommendations require approval)
max_auto_actions: 0 (recommendations only)

example_output:
  recommendation_id: rec_49284
  campaign_type: "Score Lift Track"
  current_completion_rate: 0.63
  projected_with_change: 0.71
  proposed_change: "Reorder actions: [validate_bureau, dispute_errors,
update_profile]"
  expected_roi: 8.3 # 8.3% improvement
  affected_cohort_size: 2847
  approval_required: true
```

Agent 4: Referral Impact Agent

```
text
agent_id: outcome.referral_impact.v1
name: Referral Impact Agent
agent_type: operator
risk_level: low
faces: [internal, partner]
milestone: M4
status: active

scope: |
  Identify partnership opportunities for improved outcomes,
  track referral effectiveness, measure partner impact

input_entities:
  - ConsumerOutcomeSummary (consumer progress)
  - PartnerOutcomeSummary (partner performance)
  - OutcomeTracks (historical outcomes)
  - MarketPartnerDirectory (available partners)

output_entities:
  - ReferralRecommendations (partner matches)
  - ReferralImpactSummary (partner effectiveness metrics)
  - ConversionTracks (referral → outcome paths)

required_tools:
  - get_outcome_summary
  - get_partner_directory
  - calculate_referral_fit
  - track_referral
  - measure_conversion

business_logic:
  Opportunity Detection:
    1. Identify consumers near-miss on outcomes (e.g., 10 pts from goal
score)
    2. Match to partner capabilities (e.g., debt consolidation loan)
    3. Calculate fit probability
    4. Estimate impact on Net Advantage

  Referral Execution:
    1. Create referral record with partner
    2. Track consumer through partner journey
    3. Capture conversion status

  Impact Measurement:
    1. Compare outcome before/after referral
    2. Measure partner contribution to success
    3. Calculate ROI per partnership
    4. Segment effectiveness by consumer demographics

error_handling:
  - Partner unavailable → Find alternative
```

```
  - Consumer declines → Track reason
  - No matching partner → Suggest future partnerships

guardrails:
  - Referrals require explicit consumer consent
  - No aggressive targeting of near-subprime consumers
  - Partner fairness scores must be >75%

requires_human_review: false
max_auto_actions: unlimited

example_output:
  referral_id: ref_29384
  consumer_id: cons_29384
  recommended_partner: "LoanPartner123"
  recommendation_reason: "Debt consolidation opportunity"
  estimated_net_advantage_lift: 2100 # $2100
  partner_conversion_probability: 0.65
  accepted: true
  conversion_date: "2026-02-15"
  actual_outcome: 1950 # $1950 achieved
```

ENGINE 2: RIGHTS & TRUST ENGINE

Purpose: Enforce data rights, manage consent, orchestrate disputes, monitor fairness
Core Responsibility: Privacy-first architecture, GDPR/CCPA compliance, advocacy automation
Agent Count: 5 agents
Materialized Views: mv_consent_summary, mv_dispute_summary, mv_fairness_summary, mv_audit_activity

Agent 5: Consent & Scope Assistant

```
text
agent_id: rights.consent_scope.v1
name: Consent & Scope Assistant
agent_type: assistant
risk_level: high
faces: [consumer, partner]
milestone: M4
status: active

scope: |
  Help consumers and partners understand data scopes and
  recommend minimal, appropriate permissions

input_entities:
  - DataRightsSummary (current consents)
```

```
  - PartnerProfile (requesting entity)
  - Purpose (intended use case)
  - DataTypes (schema of available data)

output_entities:
  - SuggestedScopes (minimal permissions)
  - Explanations (plain-language descriptions)
  - PrivacyImpactAssessment (risk level)

required_tools:
  - get_rights_summary
  - explain_scope
  - assess_privacy_risk
  - generate_consent_language

business_logic:
  1. Analyze partner's stated purpose
     - Validate purpose against GDPR article 6 legal bases
     - Map to minimum required data types

  2. Data Minimization Check
     - Current consents already sufficient? Return
     - Identify gap (e.g., missing employment data for underwriting)
     - Suggest minimal scope (e.g., "last 2 years employment only")

  3. Risk Assessment
     - Classify scope as low/medium/high sensitivity
     - Evaluate partner's fairness/security track record
     - Flag if high-risk scope to high-risk partner

  4. Explain & Present
     - Generate plain-language scope descriptions
     - Create visual privacy impact summary
     - Show alternatives (e.g., "We can verify income without salary
history")

error_handling:
  - Vague purpose → Request clarification
  - Impossible minimization → Explain why data needed
  - High-risk combination → Require explicit opt-in + review

guardrails:
  - Never recommend over-scoped permissions
  - Require explicit consent for sensitive data (financial, health,
location)
  - Partner fairness score must be >75%
  - Scope must align with stated purpose

requires_human_review: true (consumer must approve)
max_auto_actions: 0 (recommendations only)

example_output:
  recommendation_id: cons_98234
  partner_name: "LoanPartner123"
  purpose: "Mortgage underwriting"
```

```
  current_consent: null
  suggested_scopes:
    - data_type: "income_employment"
      fields: ["employer", "job_title", "annual_income"]
      retention_months: 24
      reason: "Required for income verification"
    - data_type: "financial_accounts"
      fields: ["bank_accounts", "balances"]
      retention_months: 6
      reason: "Liquid assets for down-payment check"
  privacy_risk: "medium"
  explanation: "LoanPartner123 needs to verify your income and assets.
We're sharing only the minimum required."
  consumer_must_review: true
```

Agent 6: Rights Request Orchestrator

```text
agent_id: rights.request_orchestrator.v1
name: Rights Request Orchestrator
agent_type: ambassador
risk_level: high
faces: [consumer, internal]
milestone: M5
status: active

scope: |
  Execute data rights requests (export, deletion, non-use)
  across all systems with validation and audit trail

input_entities:
  - RightsRequest (consumer request: type, scope)
  - DataRightsSummary (current data state)
  - AffectedSystems (systems holding data)

output_entities:
  - ConsentEvent (revocation/update)
  - AuditEvent (compliance record)
  - StatusUpdate (consumer notification)
  - ExportPackage (for data portability)

required_tools:
  - execute_rights_action
  - verify_consumer_identity
  - create_audit_event
  - enqueue_notification
  - generate_export_file

business_logic:
  Identity Verification:
    1. Verify consumer identity (multi-factor)
    2. Validate request signature (if required)

  Request Parsing:
    1. Extract request type (export, delete, restrict, portability)
```

```
    2. Identify data scope (all vs. specific partner vs. specific
purpose)
    3. Calculate effort & timeline

  System Mapping:
    1. Identify all systems holding data (primary DB, analytics, partner
systems)
    2. Check for data sharing dependencies
    3. Determine deletion order (cascade vs. independent)

  Execution (with rollback):
    1. For each system:
       a. Create backup (immutable audit snapshot)
       b. Execute action (delete/export/restrict)
       c. Verify completion
       d. Log result with timestamp
    2. If any failure:
       a. Attempt rollback for completed actions
       b. Notify support team
       c. Queue for manual completion

  Completion:
    1. Generate audit trail (immutable record)
    2. Create export package (if data portability)
    3. Send consumer confirmation
    4. Archive request with proof

error_handling:
  - Identity verification failed → Reject with reason
  - Partial failure → Rollback, require retry
  - External system timeout → Queue for manual + auto-retry
  - Deletion cascade conflict → Flag for manual review

guardrails:
  - Irreversible actions (delete) require explicit confirmation
  - Max 1 deletion per 30 days per consumer (prevent abuse)
  - Must preserve audit trail (never delete AuditEvents)
  - Notify partner systems of deletions (legal requirement)

requires_human_review: true (deletion always requires review)
max_auto_actions: 1 per 30 days (deletion only)

example_output:
  request_id: rights_39284
  consumer_id: cons_29384
  request_type: "data_deletion"
  scope: "All data for Partner123"
  status: "completed"
  systems_affected:
    - system: "primary_db"
      action: "delete"
      records_deleted: 47
      verified_at: "2026-01-15T14:23:00Z"
    - system: "partner123_crm"
      action: "notify_deletion"
```

```
      status: "notified"
  audit_trail_id: audit_49284
  consumer_notified: true
```

Agent 7: Dispute & Advocacy Agent

```
text
agent_id: rights.dispute_advocacy.v1
name: Dispute & Advocacy Agent
agent_type: ambassador
risk_level: high
faces: [consumer, internal]
milestone: M4
status: active

scope: |
  Draft credit bureau dispute letters, submit via FCRA-compliant
  channels, track resolution, recommend next steps

input_entities:
  - DisputeSummary (items to dispute)
  - AdvocacyMandate (consumer authorization)
  - EvidenceDocs (supporting documentation)
  - DisputeTemplate (letter format + legal language)

output_entities:
  - DisputeLetter (generated & verified letter)
  - PortalSubmission (bureau receipt)
  - AuditEvent (compliance record)
  - DisputeFollowUp (scheduled tracking)

required_tools:
  - draft_letter
  - validate_fcra_compliance
  - submit_dispute (bureau API or certified mail)
  - create_audit_event
  - schedule_followup
  - track_dispute_status

business_logic:
  Preparation:
    1. Verify consumer has granted advocacy rights
    2. Validate dispute items (errors, inaccuracies, unauthorized)
    3. Gather evidence (consumer statements, docs, screenshots)
    4. Prioritize disputes (high-impact first)

  Letter Generation:
    1. Select appropriate template (error correction, fraud, identity
theft, etc.)
    2. Personalize with consumer data, dispute items, evidence refs
    3. Generate plain-language summary
    4. Validate FCRA compliance (required disclosures, timelines)

  Submission:
    1. Submit to credit bureau via API (if available) or certified mail
```

```
    2. Capture submission receipt + reference number
    3. Create AuditEvent with full letter text + metadata
    4. Schedule auto-follow-up (30-day check-in)

  Tracking & Resolution:
    1. Poll dispute status (monthly)
    2. When bureau responds:
       a. Update dispute record (approved/rejected/partial)
       b. Notify consumer
       c. If approved: recalculate score
       d. If rejected: recommend escalation (attorney, ombudsman)

error_handling:
  - No advocacy rights → Inform consumer, require consent first
  - Insufficient evidence → Request additional docs
  - Invalid dispute item → Explain FCRA scope limits
  - Bureau API failure → Fall back to certified mail
  - Tracking failed → Manual follow-up queue

guardrails:
  - Must use FCRA-compliant letter template
  - Cannot make unsupported claims in dispute
  - Must verify consumer identity before submission
  - Cannot submit duplicate disputes within 30 days
  - External communications require consumer approval

requires_human_review: true (all submissions reviewed before mail)
max_auto_actions: 1 per dispute (scheduled follow-up only)

example_output:
  case_id: case_29384
  consumer_id: cons_29384
  dispute_type: "unauthorized_account"
  items:
    - tradeline_id: "account_123"
      bureau: "Equifax"
      claim: "Account opened without authorization"
  letter_generated: true
  letter_id: letter_49284
  submission_method: "api"
  submission_date: "2026-01-15"
  bureau_reference_id: "EQX-DISP-4938284"
  status: "submitted"
  followup_scheduled: "2026-02-15"
  estimated_resolution_date: "2026-02-28"
```

Agent 8: Fairness Analysis Agent

```
text
agent_id: rights.fairness_analysis.v1
name: Fairness Analysis Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M3
```

```
status: active

scope: |
  Analyze fairness metrics across demographics and segments,
  detect bias patterns, recommend investigations

input_entities:
  - FairnessSummary (aggregated fairness metrics)
  - UnderwritingSummary (decision patterns by segment)
  - PredictionOutputs (model predictions by segment)
  - ProtectedClassSegmentStats (demographic breakdowns)

output_entities:
  - FairnessReport (detailed analysis)
  - BiasDetectionAlerts (flagged issues)
  - InvestigationRecommendations (next steps)
  - RemediationStrategies (proposed fixes)

required_tools:
  - get_fairness_summary
  - compute_fairness_metrics
  - detect_disparate_impact
  - identify_bias_sources
  - propose_remediation

business_logic:
  Data Collection (weekly):
    1. Aggregate fairness metrics per engine:
       - Approval/decline rates by demographic
       - Score distributions by segment
       - Model prediction accuracy by protected class
       - Loan default rates by partner/segment

  Statistical Testing:
    1. Compute disparate impact (80/20 rule for employment law)
    2. Calculate confidence intervals per segment
    3. Test for statistical significance
    4. Model calibration (are predictions equally accurate across
groups)

  Bias Detection:
    1. Identify segments with approval disparities > threshold
    2. Trace disparity to root causes (missing data, model bias,
external factors)
    3. Calculate impact (estimated wrongful denials)
    4. Assess severity (small sample vs. systematic bias)

  Recommendations:
    1. If data quality issue: Queue data improvement campaign
    2. If model bias: Propose retraining with fairness constraints
    3. If policy bias: Recommend policy review + adjustment
    4. If external (e.g., economic): Flag for human review

  Transparency:
    1. Generate consumer-facing fairness summary
```

```
    2. Explain decision factors that may have contributed
    3. Suggest consumer actions (e.g., build history in underrepresented
segment)

error_handling:
  - Small cohort (<30) → Insufficient data, skip
  - Confounding variables → Flag for deeper analysis
  - Catastrophic drift → Immediate alert to compliance team

guardrails:
  - Cannot suppress or ignore fairness findings
  - All recommendations require compliance review
  - Changes to models/policies require fairness re-validation
  - Reporting to regulators when findings indicate violation

requires_human_review: true (all recommendations reviewed by compliance)
max_auto_actions: 0 (analysis only, recommendations require approval)

example_output:
  report_id: fair_49284
  analysis_period: "2026-01-01 to 2026-01-31"
  findings:
    - finding_id: fair_finding_1
      severity: "high"
      metric: "approval_rate_disparity"
      description: "Black/African American applicants 18% less likely to
be approved"
      affected_count: 247
      affected_pct: 12.3
      statistical_significance: "p < 0.01"
      likely_cause: "Income data missing for 40% of affected group"
      recommendation: "Implement alternative income verification (UPI,
employer API)"
  investigations_triggered: 2
  remediation_required: true
  escalation_level: "compliance_review"
```

Agent 9: Audit & Compliance Reporting Agent

```
text
agent_id: rights.compliance_reporting.v1
name: Audit & Compliance Reporting Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M5
status: active

scope: |
  Generate regulator-ready audit reports for GDPR, CCPA, FCRA,
  and SOC 2 compliance

input_entities:
  - AuditEvent (comprehensive audit log)
  - ConsentSummary (consent grants/revokes)
```

```
  - DisputeSummary (dispute resolution stats)
  - SecurityIncidentSummary (security events)

output_entities:
  - AuditReport (formatted for regulators)
  - ComplianceChecklist (certification status)
  - GapAnalysis (areas of non-compliance)

required_tools:
  - get_audit_events
  - filter_by_legal_basis
  - generate_report
  - calculate_soc2_controls
  - export_to_format

business_logic:
  Data Gathering (monthly):
    1. Query AuditEvents (all data access, modifications, consents)
    2. Aggregate by legal basis (contract, consent, legitimate interest)
    3. Filter by jurisdiction (GDPR = EU, CCPA = CA, etc.)

  Report Generation:
    1. Create GDPR Article 32 technical measures report
    2. Generate CCPA consumer rights requests summary
    3. Compile FCRA dispute handling audit
    4. Assess SOC 2 control effectiveness

  Export:
    1. Format for regulator submission
    2. Redact sensitive info appropriately
    3. Include statistical summaries

error_handling:
  - Audit log gaps → Flag and investigate
  - Missing consent records → Escalate

guardrails:
  - Must be approved by legal before submission
  - Cannot omit or minimize findings

requires_human_review: true
max_auto_actions: 0 (draft only)
```

ENGINE 3: RISK & SECURITY ENGINE

Purpose: Continuous monitoring, threat detection, incident management, security posture
Core Responsibility: Identity monitoring, breach detection, device/network profiling, remediation
Agent Count: 4 agents
Materialized Views: mv_consumer_security_summary,

mv_security_incidents, mv_platform_threat_trends

Agent 10: Security Alert Aggregator

```
text
agent_id: risk.alert_aggregator.v1
name: Security Alert Aggregator
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M3
status: active

scope: |
  Consolidate low-level security signals into actionable
  incidents with severity ranking and routing

input_entities:
  - SecurityAlert (individual alerts from multiple sources)
  - DeviceProfile (device trust history)
  - NetworkProfile (network anomalies)
  - BreachExposures (breach database matches)

output_entities:
  - SecurityIncident (consolidated incident)
  - PrioritizedAlerts (ranked by severity)
  - IncidentCase (escalation to Case Canvas)

required_tools:
  - get_alerts
  - aggregate_related_alerts
  - compute_severity
  - create_incident
  - route_to_queue

business_logic:
  1. Collect alerts from all sources (device anomaly, dark web match,
broker flag)
  2. Correlation:
     a. Group related alerts (same consumer, time window)
     b. Compute severity (single alert vs. coordinated attack pattern)
     c. Flag if escalation indicators present (credential compromise,
account takeover)
  3. Ranking:
     a. Critical: Account takeover, active fraud, identity theft
     b. High: Breach exposure, credential compromise, suspicious device
     c. Medium: New device, location anomaly, broker listing
     d. Low: Stale data, informational
  4. Routing:
     a. Critical → Immediate consumer notification + support team
     b. High → Consumer notification + suggest remediation
     c. Medium → Informational notification
     d. Low → Background monitoring only
```

```
error_handling:
  - Duplicate alerts → Deduplicate, don't escalate twice
  - False positives → Track and tune detection rules

guardrails:
  - No automated blocking (only notify)
  - Consumer opt-in for auto-remediation

requires_human_review: false
max_auto_actions: unlimited

example_output:
  incident_id: inc_39284
  consumer_id: cons_29384
  severity: "high"
  alert_count: 3
  alerts:
    - source: "dark_web_scanner"
      type: "credential_exposed"
      data_exposed: "email + password"
      breach_date: "2026-01-10"
    - source: "device_anomaly"
      type: "new_device_login"
      device: "iPhone 15 (new)"
      location: "overseas"
    - source: "broker_monitor"
      type: "listing_found"
      broker: "EquifaxPlus"
      listed_reason: "high_credit_score"
  recommendation: "Potential compromise — recommend password reset + MFA
enable"
  case_created: true
  case_id: case_49284
  notification_sent: true
```

Agent 11: Security Remediation Agent

```
text
agent_id: risk.remediation.v1
name: Security Remediation Agent
agent_type: ambassador
risk_level: high
faces: [consumer]
milestone: M4
status: active

scope: |
  Suggest and execute security remediation actions with
  consumer explicit opt-in (auto-execute not without approval)

input_entities:
  - ConsumerSecuritySummary (current security state)
  - SecurityIncident (incident details)
  - BreachExposures (what was compromised)
  - RemediationOptions (available actions)
```

```
output_entities:
  - RemediationPlan (suggested actions + effort)
  - AutomatedRequests (broker removal, freeze requests)
  - RemediationTracking (completion status)

required_tools:
  - get_security_summary
  - draft_remediation
  - send_broker_removal_letter
  - request_credit_freeze
  - send_fraud_alert
  - enqueue_remediation_task

business_logic:
  Assessment:
    1. Analyze breach exposure (what data was exposed)
    2. Assess risk to consumer (likelihood + impact)
    3. Check current security posture (freezes already in place,
monitors active)

  Recommendation:
    1. Identify relevant remediation actions:
       a. Password reset (if credentials exposed)
       b. Credit freeze (if SSN exposed)
       c. Fraud alert (if identity theft risk)
       d. Broker removal (if listing found)
       e. Monitor signup (if not already)
    2. Estimate effort per action (5 min to 2 hours)
    3. Rank by impact & urgency

  Consumer Approval:
    1. Present plan with clear explanations
    2. Get explicit opt-in per action
    3. Optionally auto-execute selected actions

  Execution:
    1. Draft letters (broker removal, fraud alert request)
    2. Submit (certified mail or portal)
    3. Track status + verify completion
    4. Update ConsumerSecuritySummary

error_handling:
  - Consumer declines → Track reason, offer again later
  - External service timeout → Retry with exponential backoff
  - Broker removal fails → Escalate to support

guardrails:
  - All actions require explicit consumer approval
  - Cannot force freeze/alert without consumer consent
  - Must present consumer with full plan before any execution

requires_human_review: false (consumer approval is gate)
max_auto_actions: unlimited (once approved)
```

```
example_output:
  remediation_plan_id: rem_49284
  consumer_id: cons_29384
  incident_id: inc_39284
  severity: "high"
  recommended_actions:
    - action_id: rem_action_1
      action_type: "password_reset"
      effort_minutes: 5
      impact: "high"
      description: "Reset password for accounts using exposed
credentials"
      recommended: true
    - action_id: rem_action_2
      action_type: "credit_freeze"
      effort_minutes: 15
      impact: "high"
      description: "Place freeze with all 3 bureaus (Equifax, Experian,
TransUnion)"
      recommended: true
    - action_id: rem_action_3
      action_type: "monitor_signup"
      effort_minutes: 10
      impact: "medium"
      description: "Enroll in monthly credit monitoring"
      recommended: false # Already enrolled
  consumer_approved_actions: [rem_action_1, rem_action_2]
  execution_status: "in_progress"
  completed: []
  failed: []
```

Agent 12: Risk Integration Agent

```
text
agent_id: risk.integration.v1
name: Risk Integration Agent
agent_type: operator
risk_level: low
faces: [internal]
milestone: M3
status: active

scope: |
  Feed security posture and risk signals into credit decisions,
  plan recommendations, and portfolio assessments

input_entities:
  - ConsumerSecuritySummary (security state)
  - RiskModel (decision thresholds)
  - UnderwritingEvent (in-progress underwriting)
  - ConsumerPlan (active plan)

output_entities:
  - RiskAdjustment (score/decision adjustment)
  - PlanModification (security-aware plan changes)
```

```
  - PortfolioRiskAlert (concentration risk)

required_tools:
  - get_security_summary
  - adjust_underwriting_decision
  - modify_plan
  - flag_portfolio_risk

business_logic:
  Integration Points:
    1. In underwriting:
       a. High risk consumer → Tighter verification, conditional
approval
       b. Compromised account → Hold decision until remediation complete
    2. In planning:
       a. Active incident → Hold automated action execution
       b. High risk → Skip high-impact actions pending resolution
    3. In portfolio:
       a. Concentration of high-risk consumers → Flag to capital team

error_handling:
  - Missing security data → Use conservative estimate
  - Conflicting signals → Escalate to underwriting review

guardrails:
  - Cannot outright deny based on security alone
  - Must document risk adjustments
  - Consumer must be notified of security-based decisions

requires_human_review: false (integration is algorithmic)
max_auto_actions: unlimited
```

Agent 13: Threat Intelligence Agent

```
text
agent_id: risk.threat_intel.v1
name: Threat Intelligence Agent
agent_type: operator
risk_level: low
faces: [internal]
milestone: M3
status: active

scope: |
  Analyze platform-wide threat trends and recommend
  preventive campaigns

input_entities:
  - PlatformThreatSummary (aggregated threat data)
  - SecurityIncidentSummary (incident trends)
  - ConsumerSegmentStats (risk by segment)

output_entities:
  - ThreatReport (findings)
  - SecurityCampaignRecommendations (proactive campaigns)
```

```
required_tools:
  - get_threat_summary
  - detect_trends
  - draft_campaigns

business_logic:
  1. Analyze threat landscape (breach frequency, attack types)
  2. Identify emerging risks (new malware, identity theft trends)
  3. Segment vulnerable populations
  4. Recommend campaigns (awareness, monitoring, remediation)

error_handling:
  - Insufficient data → Use external threat intelligence

guardrails:
  - Cannot trigger campaigns automatically (must be approved)

requires_human_review: true
max_auto_actions: 0 (recommendations only)
```

ENGINE 4: MARKET & CAPITAL ENGINE

Purpose: Turn consumer data into ethical loan products, manage underwriting and capital markets
Core Responsibility: Ingestion, underwriting, QC, packaging, lender portfolio management
Agent Count: 5 agents
Materialized Views: mv_ops_ingestion_summary, mv_underwriting_summary, mv_loan_package_performance, mv_qc_summary

Agent 14: Ingestion Mapping Agent

```
text
agent_id: market.ingestion_mapping.v1
name: Ingestion Mapping Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M3
status: active

scope: |
  Auto-map partner data payloads to canonical Loan schema,
  identify errors, suggest schema updates

input_entities:
  - PartnerPayload (raw partner data)
  - LoanSchema (canonical schema)
```

```
  - PartnerMappingHistory (prior mapping attempts)

output_entities:
  - MappingSuggestions (auto or manual)
  - ErrorClusters (data quality issues)
  - SchemaUpdateRecommendations (if canonical schema too narrow)

required_tools:
  - analyze_payload
  - infer_mapping
  - suggest_mapping
  - cluster_errors
  - validate_schema

business_logic:
  1. Parse partner payload (JSON, CSV, XML)
  2. Infer field types and meanings (ML-assisted)
  3. Match to canonical schema fields
  4. Handle mismatches:
     a. Type mismatch → suggest transformation
     b. New field → suggest addition to schema
     c. Missing field → flag as error
  5. Validate mappings:
     a. Type validation
     b. Range validation (loan amounts, terms)
     c. Business rule validation (rate must be > 0)
  6. Cluster errors for bulk fixes

error_handling:
  - Ambiguous field → Request clarification from partner
  - New fields → Propose schema extension
  - Systemic errors → Pause ingestion, notify partner

guardrails:
  - Cannot drop required fields
  - Mapping changes require approval before deployment
  - Must validate after schema changes

requires_human_review: true (new mappings require approval)
max_auto_actions: 0 (recommendations only)

example_output:
  mapping_id: map_49284
  partner_id: partner_123
  payload_sample_count: 1000
  status: "needs_review"
  mapping_results:
    - partner_field: "loan_amount_cents"
      canonical_field: "principal"
      transformation: "divide_by_100"
      confidence: 0.98
    - partner_field: "annual_interest_rate_bp"
      canonical_field: "rate"
      transformation: "divide_by_10000"
      confidence: 0.95
```

```
    - partner_field: "origination_time"
      canonical_field: "originated_at"
      transformation: "parse_iso8601"
      confidence: 0.99
  errors_detected:
    - error_type: "missing_required"
      field: "term_months"
      affected_count: 47
      pct: 4.7
    - error_type: "invalid_range"
      field: "rate"
      affected_count: 12
      range_observed: "-0.5 to 25.0"
      expected_range: "0.0 to 20.0"
  recommendation: "Fix term_months for 47 records; clarify rate
definition with partner"
  requires_approval: true
```

Agent 15: Underwriting Decision Agent

```
text
agent_id: market.underwriting_decision.v1
name: Underwriting Decision Agent
agent_type: ambassador
risk_level: high
faces: [internal, partner]
milestone: M4
status: active

scope: |
  Draft underwriting decisions (approve, deny, conditional)
  with FCRA-compliant reason codes and adverse action notices

input_entities:
  - LoanApplication (application + consumer data)
  - UnderwritingCriteria (decision rules)
  - CreditProfile (credit bureau data)
  - ConsumerOutcomeSummary (Net Advantage, plan status)
  - FairnessSummary (fairness constraints)

output_entities:
  - UnderwritingEvent (decision record)
  - Decision (approve/deny/conditional)
  - ReasonCodes (FCRA Article V reason codes)
  - AdverseActionNotice (if denied/conditional)
  - AuditEvent (full decision trail)

required_tools:
  - get_application
  - get_credit_profile
  - evaluate_application
  - check_fairness_constraints
  - draft_decision
  - generate_adverse_action_notice
  - create_audit_event
```

```
business_logic:
  Data Gathering:
    1. Fetch loan application (amount, term, purpose)
    2. Fetch consumer credit profile (FICO, recent inquiries, accounts)
    3. Fetch outcome + fairness context

  Evaluation (multi-stage):
    Stage 1 — Rule-based checks:
      a. Income verification (DTI < 45%)
      b. Credit score (min FICO 580, or alternative data)
      c. Account history (min 2 years established credit)
      d. Delinquency check (no recent 30+ day lates)

    Stage 2 — Model-based decision:
      a. Feed approved attributes to underwriting model
      b. Generate risk probability + confidence interval
      c. Apply decision rule (e.g., approve if P(default) < 5%)
      d. Generate reason codes from model feature importance

    Stage 3 — Fairness validation:
      a. Check if decision would create disparate impact
      b. If so: flag for manual review (do not auto-deny)

  Output Generation:
    1. Decision (approve/deny/conditional)
    2. Reason codes (FCRA Article V compliant):
       - "Income insufficient relative to loan amount"
       - "Credit file too new to determine creditworthiness"
       - "Specific account showing delinquency or default"
    3. Adverse action notice (if deny/conditional):
       - Decision summary
       - Reason codes
       - Consumer rights
       - FCRA disclosure
    4. Terms (if approved):
       - Loan amount, rate, term
       - Conditions (if any)

error_handling:
  - Missing data → Request from consumer/partner
  - Edge cases (unusual income types) → Flag for manual review
  - Fairness flag → Always escalate (do not auto-deny)
  - Model error → Use baseline conservative decision

guardrails:
  - All deny/conditional decisions require adverse action notice
  - Cannot make decision if material data missing
  - Must document reason codes
  - Fairness constraints must be checked pre-decision
  - Consumer has right to know adverse action reasons

requires_human_review: true (flagged cases only: fairness, edge cases)
max_auto_actions: unlimited (routine approvals auto-execute)
```

```
example_output:
  underwriting_event_id: uw_49284
  loan_id: loan_29384
  consumer_id: cons_29384
  application_amount: 25000
  decision: "approved"
  approved_amount: 25000
  approved_rate: 8.5
  approved_term_months: 60
  reason_codes:
    - "Monthly income sufficient for loan payment obligation"
    - "Credit history demonstrates responsible credit management"
  fairness_check_passed: true
  adverse_action_required: false
  confidence_score: 0.92
  model_version: "uw_model_v3.2"
  created_at: "2026-01-15T14:23:00Z"
```

Agent 16: QC Review Scheduler Agent

```
text
agent_id: market.qc_scheduling.v1
name: QC Review Scheduler Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M4
status: active

scope: |
  Schedule QC reviews on loan decisions with risk-based
  sampling and exception flagging

input_entities:
  - UnderwritingEvent (decisions to review)
  - QCRules (sampling rules)
  - RiskAssessment (decision risk scores)

output_entities:
  - QCReviewTask (scheduled review)
  - ReviewQueue (prioritized for QC team)
  - HighRiskFlag (auto-escalation)

required_tools:
  - get_underwriting_events
  - calculate_sample_size
  - identify_high_risk
  - create_qc_task
  - enqueue_for_review

business_logic:
  Sampling Strategy:
    1. Risk-based stratified sampling:
       - High-risk decisions (fairness flags, edge cases): 100% review
       - Medium-risk (conditional approvals): 20% review
```

```
      - Low-risk (routine approvals): 5% review
    2. Statistical sampling (maintain power for error detection)

  Exception Flagging:
    1. Auto-escalate for immediate review:
       a. Fairness flags
       b. Model errors
       c. Unusual reason code combinations

  Task Creation:
    1. Create QCReview task with:
       - Decision details
       - Reason codes
       - Risk flags
       - Reviewer guidance

error_handling:
  - Insufficient samples → Recommend sampling rate adjustment

guardrails:
  - Cannot skip high-risk reviews
  - Must document sampling methodology

requires_human_review: false
max_auto_actions: unlimited
```

Agent 17: Packaging Optimization Agent

```
text
agent_id: market.packaging_optimization.v1
name: Packaging Optimization Agent
agent_type: operator
risk_level: high
faces: [internal]
milestone: M4
status: active

scope: |
  Optimize loan grouping for secondary market sale,
  balancing yield, diversification, and risk

input_entities:
  - LoanPool (available loans for sale)
  - BuyerConstraints (target buyer requirements)
  - MarketConditions (pricing, demand)
  - PortfolioRiskLimits (internal risk tolerance)

output_entities:
  - ProposedLoanPackages (optimized groupings)
  - ExpectedYield (projected returns)
  - RiskAssessment (package risk profile)
  - DiversificationMetrics (concentration risks)

required_tools:
  - get_loan_pool
```

```
  - parse_buyer_constraints
  - optimize_package
  - validate_constraints
  - compute_yield
  - simulate_scenarios
  - analyze_diversification

business_logic:
  Constraint Parsing:
    1. Extract buyer requirements:
       a. Geography (states allowed)
       b. Credit score ranges (FICO bands)
       c. Loan amount range
       d. Term range
       e. Rate caps/floors
       f. Maximum concentration (% per state, income level, etc.)

  Optimization:
    1. Define objective function:
       - Maximize yield (weighted by risk-adjusted returns)
       - Subject to constraints (buyer + internal risk limits)
    2. Run optimization algorithm:
       - Select loans meeting buyer constraints
       - Group to maximize yield while maintaining diversification
       - Avoid concentration risks
       - Ensure fairness balance (diverse demographics)
    3. Generate 3 proposals (conservative, balanced, aggressive)

  Scenario Simulation:
    1. Model market scenarios:
       a. Interest rate change (±2%)
       b. Prepayment rates
       c. Default rates
    2. Calculate impact on yield + loss reserves
    3. Stress-test against portfolio risk limits

  Output:
    1. Package 1 (Conservative):
       - Lower yield, very low risk
       - Broadest buyer appeal
    2. Package 2 (Balanced):
       - Medium yield, medium risk
       - Recommended
    3. Package 3 (Aggressive):
       - Higher yield, higher risk
       - For sophisticated investors

error_handling:
  - Insufficient inventory → Notify sales
  - Conflicting constraints → Flag as impossible, suggest adjustments
  - Optimization failure → Use greedy selection

guardrails:
  - Cannot violate buyer constraints
  - Must maintain internal risk limits
```

```
  - Must validate diversification
  - Fairness must be reviewed (not disparate in outcome)

requires_human_review: true (must review before trade)
max_auto_actions: 0 (recommendations only)

example_output:
  package_proposal_id: pkg_49284
  buyer_name: "National Lender Corp"
  constraint_summary: "FICO > 650, 30-year loans, rates < 6.5%"
  loan_pool_size: 1247
  eligible_loans: 384
  packages_proposed: 3
  package_1:
    name: "Conservative NLC-001"
    loan_count: 95
    total_principal: 2847300
    weighted_avg_rate: 5.2
    expected_yield: 3.1 # percent per year
    weighted_avg_fico: 723
    geographic_diversity: 38 states
    concentration_risk: "low"
  package_2:
    name: "Balanced NLC-002"
    loan_count: 127
    total_principal: 3947200
    weighted_avg_rate: 5.8
    expected_yield: 4.2
    weighted_avg_fico: 698
    geographic_diversity: 45 states
    concentration_risk: "medium"
    recommendation: "RECOMMENDED"
  recommendation_summary: "NLC-002 balances yield and risk within all
constraints"
  requires_approval: true
  approval_deadline: "2026-01-17"
```

Agent 18: Portfolio Risk Monitor Agent

```
text
agent_id: market.portfolio_risk_monitor.v1
name: Portfolio Risk Monitor Agent
agent_type: operator
risk_level: medium
faces: [internal]
milestone: M4
status: active

scope: |
  Monitor loan portfolio health and flag concentration risks,
  performance deterioration, or model drift

input_entities:
  - LoanPortfolio (all active loans per lender)
  - PortfolioPerformanceMetrics (historical + current)
```

```
  - ModelPredictions (original underwriting predictions)

output_entities:
  - PortfolioHealthAlert (risk flags)
  - DriftAnalysis (prediction vs. reality)
  - ConcentrationRiskReport (exposure risks)
  - RemediationRecommendations (potential fixes)

required_tools:
  - get_portfolio_metrics
  - compute_default_rate
  - detect_concentration
  - analyze_drift
  - forecast_loss

business_logic:
  Health Monitoring:
    1. Track key metrics:
       a. Actual vs. predicted default rate
       b. Loss rate by cohort
       c. Prepayment speeds
    2. Detect deterioration:
       a. Trending worse? Flag early
       b. Segment underperformance (e.g., specific partner origin)
    3. Alert if SLAs missed

  Concentration Risk:
    1. Measure exposures:
       a. Geographic concentration
       b. Lender concentration
       c. Originator concentration
       d. Demographic concentration
    2. Flag if concentration > limits

  Model Drift:
    1. Compare actual vs. predicted defaults
    2. If drift detected:
       a. Identify root cause (economic changes, underwriting drift)
       b. Recommend retraining or decision threshold adjustment

error_handling:
  - Data gaps → Use last-known value

guardrails:
  - Cannot adjust prices retroactively
  - Must document concentration findings
  - Recommendations require trading desk review

requires_human_review: false (monitoring is algorithmic)
max_auto_actions: unlimited (alerts only)
```

CROSS-CUTTING AGENTS

Purpose: Serve all engines and all faces with horizontal

capabilities
Agent Count: 2 agents

Agent 19: Explainer Agent

```
text
agent_id: cross.explainer.v1
name: Explainer Agent
agent_type: assistant
risk_level: low
faces: [consumer, partner, internal]
milestone: M3
status: active

scope: |
  Generate human-readable explanations for automated decisions,
  predictions, and recommendations

input_entities:
  - Any entity (ConsumerPlan, Decision, Prediction, etc.)
  - DecisionContext (what, who, why)
  - ExplanationTemplate (format preferences)

output_entities:
  - Explanation (plain-language text)
  - FactorBreakdown (importance of contributing factors)
  - NextSteps (what to do with this info)
  - Transparency (fairness + bias notes)

required_tools:
  - get_entity_context
  - generate_explanation
  - compute_factor_importance
  - generate_transparency_note

business_logic:
  1. Analyze entity being explained (e.g., loan decision)
  2. Extract key factors contributing to outcome
  3. Rank by importance (feature importance from ML models)
  4. Generate plain-language narrative:
     - What happened (decision/outcome)
     - Why it happened (top 3-5 factors)
     - What you can do about it (next steps)
  5. Add fairness transparency:
     - Note if decision was affected by protected class (if any)
     - Provide recourse (dispute, appeal process)

error_handling:
  - Complex decision → Simplify to core factors
  - Model error → Use baseline explanation

guardrails:
  - Cannot explain in jargon
```

```
  - Must be truthful (no sugar-coating risks)
  - Must include fairness disclosures where relevant

requires_human_review: false
max_auto_actions: unlimited (read-only)

example_output:
  explanation_id: exp_49284
  subject: "Your creditX Score"
  subject_type: "score"
  summary: "Your creditX score is 695, up 15 points in the last 90
days."
  narrative: |
    Here's what's helping your score:
    1. Payment history: All on-time payments in the last 3 months (+12
points)
    2. Account diversity: You now have a credit mix (credit cards +
installment loan) (+8 points)

    Here's what could improve it more:
    1. Utilization: Your credit cards are at 42% of your limits (lower
is better — try to get below 30%)
    2. Age of accounts: Your newest account is 4 months old (older
history helps, so keep those old accounts open)

  factor_breakdown:
    - factor: "Payment history"
      importance: 0.35
      direction: "positive"
      contribution_points: 12
    - factor: "Account diversity"
      importance: 0.25
      direction: "positive"
      contribution_points: 8
    - factor: "Utilization ratio"
      importance: 0.20
      direction: "negative"
      contribution_points: -5

  next_steps:
    - "Target: Get utilization below 30%"
    - "Action: Pay down balances before statement closing date"
    - "Estimated impact: +8 points in 30 days"

  fairness_note: "This score was calculated without regard to age, race,
religion, sex, marital status, or national origin, per FCRA."
```

Agent 20: Notification Agent

```
text
agent_id: cross.notification.v1
name: Notification Agent
agent_type: operator
risk_level: low
faces: [consumer, partner, internal]
```

```
milestone: M2
status: active

scope: |
  Queue notifications based on events, with delivery preference
  and timing optimization

input_entities:
  - EngineEvent (any event from any engine)
  - NotificationPreference (consumer/partner preferences)
  - DeliveryChannel (SMS, email, push, in-app)

output_entities:
  - QueuedNotification (enqueued for delivery)
  - DeliveryLog (delivery record)

required_tools:
  - enqueue_notification
  - filter_by_preference
  - schedule_delivery
  - log_delivery

business_logic:
  Event Detection:
    1. Listen for events from all engines:
       - Plan milestone achieved
       - Alert triggered
       - Decision made
       - Action completed

  Preference Filtering:
    1. Check consumer/partner notification settings:
       - Notification type enabled?
       - Preferred channel(s)?
       - Quiet hours (don't notify 9pm-8am)?
    2. Skip if preference disabled

  Queue Management:
    1. Enqueue notification with:
       - Recipient, type, message template, context
       - Delivery preference (channel, timing)
    2. Deduplicate (don't send same notification twice)

  Batch Delivery:
    1. Deliver in batches (hourly, end-of-day)
    2. Prioritize urgent notifications
    3. Log all deliveries (audit trail)

error_handling:
  - Delivery failure → Retry with exponential backoff
  - Invalid contact → Mark and skip

guardrails:
  - Respect notification preferences (no spam)
  - Respect regulatory limits (e.g., no SMS for minors)
```

```
  - Never send sensitive info to unverified contacts

requires_human_review: false
max_auto_actions: unlimited

example_output:
  notification_id: notif_49284
  consumer_id: cons_29384
  event_type: "plan_milestone_achieved"
  event_context:
    milestone: "50% progress to score goal"
    score_improvement: 25
    days_elapsed: 45
  message_template: "plan_milestone_v1"
  message_rendered: |
    🎉 Halfway there! Your score has improved by 25 points in 45 days.
    Keep up the momentum - you're on track to hit your goal!
  delivery_preferences:
    channels: [email, in_app, push]
    send_at: "end_of_day"
    quiet_hours: [21, 8] # 9pm - 8am
  status: "queued"
  queued_at: "2026-01-15T14:23:00Z"
  scheduled_delivery: "2026-01-15T18:00:00Z" # end-of-day
```

Agent Governance & Deployment

Registry Standards

All agents live in the **Agent Registry** with versioned configs:

```
text
# agents/outcome/plan_generation.v1.yaml
agent_id: outcome.plan_generation.v1
version: 1
status: active
milestone: M3
owner_team: outcome_engineering

# agents/market/underwriting_decision.v2.yaml (updated version)
agent_id: market.underwriting_decision.v2
version: 2
status: active
milestone: M4
deprecates: outcome.underwriting_decision.v1
changelog:
  - "Added fairness constraint validation"
  - "Updated FCRA reason code mappings"
  - "Improved edge case handling"
```

Deployment Process

1. **Code Review** - Engineering review of agent implementation

2. **Fairness & Compliance Review** - Legal + Risk review
3. **Testing** - Unit, integration, end-to-end tests
4. **Staging Validation** - Dry-run in sandbox
5. **Gradual Rollout** - Feature flag release (1% → 10% → 50% → 100%)
6. **Monitoring** - Track agent performance metrics + incidents
7. **Production Hotline** - Support team escalation path

Monitoring & Observability

Each agent tracks:

```
text
metrics:
  - invocations_per_day
  - success_rate_pct (exclude human_override_rate)
  - avg_latency_ms
  - p95_latency_ms
  - error_rate_pct
  - human_override_rate_pct
  - cost_per_invocation (LLM API calls + compute)

alerts:
  - success_rate < 95% → Page on-call
  - latency p95 > SLA → Investigate
  - error_rate > 1% → Auto-disable or roll back
  - human_override_rate > threshold → Review implementation
  - fairness metrics degraded → Escalate to compliance
```

Build Roadmap & Milestones

Phase Map: Agent Implementation Order

M0-M1 (Foundation, Wclass

- NotificationAgent (cross-cutting, low-complexity, unlocks others)
- SecurityAlertAggregator (risk foundation, essential for security posture)
- ExplainerAgent (universal, needed for all consumer interactions)

M2-M3 (Core Engines, Mclass)

- PlanGenerationAgent (Outcome, critical for consumer OS)
- OutcomeEvaluationAgent (Outcome, tracks progress)
- ThreatIntelAgent (Risk, proactive security)
- FairnessAnalysisAgent (Rights, compliance critical)
- IngestionMappingAgent (Market, data foundation)

M3-M4 (Rights & Decisions, Mclass)

- ConsentScopeAssistant (Rights, privacy-first architecture)
- DisputeAdvocacyAgent (Rights, consumer advocacy)
- SecurityRemediationAgent (Risk, consumer empowerment)
- UnderwritingDecisionAgent (Market, lending core)
- RiskIntegrationAgent (Risk, cross-engine integration)
- CampaignTuningAgent (Outcome, optimization)
- ProgressTrackerAgent (Outcome, consumer engagement)
- ReferralImpactAgent (Outcome, partnerships)

M4-M5 (Advanced, Mclass)

- RightsRequestOrchestrator (Rights, complex workflows)
- PackagingOptimizationAgent (Market, capital markets)
- AuditComplianceReportingAgent (Rights, regulatory)
- PortfolioRiskMonitorAgent (Market, portfolio management)
- PredictionCalibrationAgent (Outcome, model governance)