Decision Logic Builder

The Decision Logic Builder is the specification for encoding expert decision-making into executable branch logic. It takes the classified task units from e4-task-decomposition-framework — specifically all units classified judgment and any deterministic units with conditional branches — and converts them into explicit decision trees with named conditions, branch edges, escalation points, stop conditions, and exception handlers. Every escalation point produced here references a threshold defined in s1-threshold-escalation-spec, ensuring that the decision model never introduces new trigger conditions outside the established reliability policy. The output feeds directly into e4-agent-role-specification (which assigns owners to branches) and e4-workflow-artifacts (which instantiates the process map and I/O contracts).

Decision Node Schema

Each decision-carrying unit in e4-task-decomposition-framework becomes one decision node. Linear deterministic units with no branches do not need a node; they appear in the process map as pass-through steps.

```
decision_nodes:
  - node_id: <kebab-case; typically matches unit_id from task-decomposition>
    unit_ref: <unit_id from e4-task-decomposition-framework>
    name: <short title>
    description: <one sentence; what is being decided>
    decision_type: <branch | gate | exception | stop>
    conditions:
      - condition_id: <kebab-case>
        label: <human-readable label>
        expression: <formal or pseudocode expression>
        confidence_band_required: <high | medium | low | any>
        branch_to: <next node_id or terminal>
    default_branch:
      branch_to: <node_id or terminal>
      requires_hitl: <true | false>
      rationale: <why this is the default>
    escalation_points:
      - trigger: <condition or combined condition expression>
        threshold_ref: <id or section from s1-threshold-escalation-spec>
        escalation_type: <hitl-gate | fallback | stop>
        escalation_target_role: <role>
        payload: <what is passed to the escalation handler>
    stop_conditions:
      - stop_id: <kebab-case>
        trigger: <condition expression>
        reason: <why the workflow terminates here>
        recovery_path: <what happens next>
    exception_handlers:
      - exception_id: <kebab-case>
        failure_mode_ref: <failure_id from s1-failure-mode-register>
        trigger: <detectable_signal expression>
        response: <agent action on detection>
        branch_to: <node_id, fallback terminal, or escalation>
```

Decision Type Definitions

branch — A node where the workflow takes one of two or more mutually exclusive paths based on evaluated conditions. All branch conditions together must be exhaustive (the default branch covers the residual case). Every

branch edge is named.

gate — A node where the workflow may only proceed if a specific condition is satisfied. Failed gates route to escalation or stop; they do not silently pass. HITL gates are gate nodes with requires_hitl: true.

exception — A node that fires only when a detectable failure-mode signal is observed mid-execution. Exception nodes interrupt the normal flow; they are not part of the happy path but must be explicitly modeled. Reference the applicable failure_id from s1-failure-mode-register.

stop — A terminal node that ends the workflow without producing a final output in the normal sense. Stop conditions must produce a structured event (error record, escalation, or provenance record) rather than silent termination.

Escalation Point Design Rules

Every escalation point must:

1. Name the threshold_ref in s1-threshold-escalation-spec that defines the trigger level. Do not introduce numeric thresholds in this spec that are not already defined there.

2. Specify the escalation_type: hitl-gate (workflow pauses for human approval), fallback (workflow switches to a declared safe alternative), or stop (workflow terminates with a structured event).

3. Declare the escalation_target_role matching a role in s1-operating-context-canvas escalation_targets.

4. Specify the payload — what information the escalation handler receives. Never escalate without a structured payload; an unstructured "something went wrong" is not an escalation.

Stop Condition Rules

A stop condition must be modeled explicitly whenever:

- The required context for a unit cannot be retrieved or meets neither the confidence_band_required nor the freshness_budget_hours threshold
- A critical failure mode signal is detected at any node
- An unsafe-to-automate unit is attempted without the required HITL gate having fired
- The workflow has exceeded its maximum retry budget (reference the retry cap in s1-threshold-escalation-spec)

Stop conditions produce a provenance record (see C5 in e4-workflow-artifacts), log the stop reason, and notify the appropriate escalation target.

Exception Handling Rules

Exception nodes are distinct from branch nodes. They represent the detection of a failure mode mid-execution and are wired off the normal flow, not as branches of normal decision logic. Every judgment unit from e4-task-decomposition-framework whose failure_modes list is non-empty must have a corresponding exception node or exception handler in the decision tree. Exceptions that cannot be handled locally (i.e., where the agent cannot

produce a safe fallback) must terminate via a stop condition.

Worked Example: South Walton Stormwater Permit Triage — Decision Tree

The following illustrates a completed decision logic build for the permit triage workflow. All figures are illustrative.

```
decision_nodes:

  - node_id: completeness-gate
    unit_ref: completeness-check
    name: Application Completeness Gate
    description: >
      Block routing until all required fields are confirmed present per
      permit-application-v3.json schema.
    decision_type: gate
    conditions:
      - condition_id: complete
        label: All required fields present and schema valid
        expression: schema_validation.passed == true AND required_fields_count ==
expected_fields_count
        confidence_band_required: high
        branch_to: wetlands-proximity-decision
      - condition_id: incomplete
        label: One or more required fields missing or invalid
        expression: schema_validation.passed == false OR required_fields_count <
expected_fields_count
        confidence_band_required: any
        branch_to: return-to-applicant-draft
    default_branch:
      branch_to: return-to-applicant-draft
      requires_hitl: false
      rationale: >
        An incomplete application cannot be routed; return-to-applicant is safe
        and reversible.
    escalation_points: []
    stop_conditions:
      - stop_id: schema-drift-stop
        trigger: >
          schema_validation.error_type == "unexpected-field" OR
          schema_validation.error_type == "type-mismatch-on-required"
        reason: >
          Schema drift on required fields indicates a portal change; the agent
          cannot safely coerce the input.
        recovery_path: >
          quarantine-input -> escalate-to-schema-owner ->
          schema-update-task (s1-failure-mode-register: schema-drift-input)
    exception_handlers:
      - exception_id: portal-timeout-on-schema-fetch
        failure_mode_ref: tool-timeout
        trigger: portal_connector.duration_ms > portal_connector.timeout_ms
        response: >
          Cancel call, mark step as degraded, attempt cached schema version
          within freshness budget; if cache expired, stop.
        branch_to: schema-drift-stop

  - node_id: wetlands-proximity-decision
    unit_ref: wetlands-proximity-check
    name: Wetlands Proximity Branch
    description: >
      Assign wetlands proximity flag based on parcel centroid distance to
      mapped wetlands boundary, subject to data freshness and confidence.
    decision_type: branch
    conditions:
```

```
      - condition_id: clear
        label: Parcel outside regulated proximity; overlay data fresh
        expression: >
          parcel_to_wetlands_distance_ft > regulated_threshold_ft AND
          fwc_overlay.age_hours <= 168 AND
          fwc_overlay.confidence_band in {high, medium}
        confidence_band_required: medium
        branch_to: routing-decision
      - condition_id: within-proximity
        label: Parcel within regulated proximity
        expression: >
          parcel_to_wetlands_distance_ft <= regulated_threshold_ft AND
          fwc_overlay.confidence_band in {high, medium}
        confidence_band_required: medium
        branch_to: engineer-review-escalation
      - condition_id: stale-or-low-confidence
        label: Overlay data stale or low confidence
        expression: >
          fwc_overlay.age_hours > 168 OR
          fwc_overlay.confidence_band in {low, unknown}
        confidence_band_required: any
        branch_to: wetlands-data-quality-escalation
    default_branch:
      branch_to: wetlands-data-quality-escalation
      requires_hitl: true
      rationale: >
        Unclassified wetlands proximity defaults to engineer escalation;
        do not route to clerk-queue without confirmed clear determination.
    escalation_points:
      - trigger: fwc_overlay.confidence_band in {low, unknown} OR fwc_overlay.age_hours >
168
        threshold_ref: s1-threshold-escalation-spec (confidence-band-threshold)
        escalation_type: hitl-gate
        escalation_target_role: District Engineer
        payload: >
          {parcel_id, fwc_overlay.age_hours, fwc_overlay.confidence_band,
          parcel_to_wetlands_distance_ft, application_id}
    stop_conditions: []
    exception_handlers:
      - exception_id: gis-connector-stale
        failure_mode_ref: stale-data
        trigger: parcel_gis_layer.age_hours > freshness_budget_hours
        response: >
          Force refresh of parcel-gis-layer; if refresh fails, demote
          confidence_band and re-evaluate branch condition.
        branch_to: wetlands-data-quality-escalation

  - node_id: routing-decision
    unit_ref: permit-routing-decision
    name: Final Routing Classification
    description: >
      Assign the application to clerk-queue, engineer-review, or
      return-to-applicant based on all collected flags.
    decision_type: branch
    conditions:
      - condition_id: engineer-review
        label: Wetlands flag set or parcel class requires professional review
        expression: >
          wetlands_flag == true OR parcel_class in {Class-III, Class-IV}
        confidence_band_required: medium
        branch_to: engineer-review-queue
      - condition_id: return-to-applicant
        label: Completeness gap or citation deficiency
        expression: completeness_flag == false OR citation_deficiency == true
        confidence_band_required: any
        branch_to: return-to-applicant-draft
      - condition_id: clerk-queue
        label: Complete, no wetlands flag, standard parcel class
        expression: >
```

```
          completeness_flag == true AND wetlands_flag == false AND
          parcel_class not in {Class-III, Class-IV}
        confidence_band_required: medium
        branch_to: clerk-queue-assignment
    default_branch:
      branch_to: engineer-review-queue
      requires_hitl: false
      rationale: >
        When routing is ambiguous, engineer review is safer than clerk-queue;
        over-routing to engineers is a medium-severity degradation, not a
        critical failure.
    escalation_points:
      - trigger: combined_confidence_band == low OR combined_confidence_band == unknown
        threshold_ref: s1-threshold-escalation-spec (confidence-band-threshold)
        escalation_type: hitl-gate
        escalation_target_role: District Engineer
        payload: >
          {application_id, all_flags, confidence_band_per_source,
          top_two_candidate_routes, rationale}
    stop_conditions:
      - stop_id: null-routing-result
        trigger: routing_decision_result == null OR all_conditions_unmatched == true
        reason: No condition evaluated to true and default branch is ambiguous.
        recovery_path: escalate-to-district-engineer -> manual-routing
    exception_handlers:
      - exception_id: low-confidence-route-detected
        failure_mode_ref: low-confidence-routing
        trigger: routing_confidence < required_confidence_threshold
        response: >
          Escalate to HITL queue with top two candidate routes and all
          contributing evidence; do not emit a routing recommendation.
        branch_to: engineer-review-escalation

  - node_id: notification-send-gate
    unit_ref: applicant-notification-send
    name: Applicant Notification HITL Gate
    description: >
      Block direct applicant communication until a district clerk has
      explicitly reviewed and approved the notification draft.
    decision_type: gate
    conditions:
      - condition_id: clerk-approved
        label: Clerk has reviewed and approved the notification draft
        expression: clerk_approval_status == "approved"
        confidence_band_required: high
        branch_to: send-notification-terminal
      - condition_id: clerk-rejected
        label: Clerk has rejected or modified the draft
        expression: clerk_approval_status in {rejected, modified}
        confidence_band_required: high
        branch_to: notification-draft-revision
    default_branch:
      branch_to: hold-for-clerk-approval
      requires_hitl: true
      rationale: >
        Default is always to hold; the gate fires only when clerk approval
        is explicitly recorded. Silence is not approval.
    escalation_points:
      - trigger: clerk_review_sla_exceeded == true
        threshold_ref: s1-threshold-escalation-spec (sla-breach-threshold)
        escalation_type: hitl-gate
        escalation_target_role: district-intake-clerk
        payload: "{application_id, draft_id, age_hours, sla_minutes_remaining}"
    stop_conditions:
      - stop_id: unsafe-send-attempted
        trigger: send_attempted == true AND clerk_approval_status != "approved"
        reason: >
          Agent attempted to send without clerk approval; hard stop per
          irreversible-impact boundary.
```

```
        recovery_path: >
          terminate-run -> emit unsafe-action-attempted event
          (s1-failure-mode-register) -> HITL gate -> postmortem
    exception_handlers:
      - exception_id: unsafe-action-intercepted
        failure_mode_ref: unsafe-action-attempted
        trigger: planned_action == "send-applicant-notification" AND clerk_approval_status
!= "approved"
        response: >
          Hard stop. Refuse the action. Emit override event with
          decision_origin=fallback. Do not attempt a softer variant.
        branch_to: unsafe-send-attempted
```

Usage Notes

Decision nodes are consumed by e4-agent-role-specification to assign agent or human ownership to each branch. HITL gates in this spec are the inputs to the HITL gate inventory in that artifact. e4-workflow-artifacts uses this spec to build the annotated process map and to confirm that every branch edge has a declared provenance emission (C5). Any branch not covered by a named condition and not covered by the default branch is a spec defect — the process map must not contain an implicit "fall through" edge.