Workflow Package Templates

The Workflow Package Templates provide six fillable scaffolds that together constitute a complete deliverable package for a new agent workflow engagement. They are the practitioner-facing output of Course 4: a workflow model, a decision map, a handoff policy, an agent task spec, an exception register, and a workflow test plan. Each scaffold is self-contained and begins with a one-line purpose statement so that a reader who opens any single document understands its role without reading the others. Fill these in order — the workflow model first, then the decision map, then the handoff policy and agent task spec in parallel, then the exception register, and finally the test plan. Pass the completed package to downstream courses (Course 7 GovTech uses e4-workflow-artifacts directly; Course 10 Commercialization uses the package to define productizable workflows).

Scaffold 1 — Workflow Model

Purpose: The directed model of all steps, decisions, branches, and handoffs in one workflow, derived from discovery and validated against expert practice.

```
# WORKFLOW MODEL
# Fill one instance per workflow. Version alongside the workflow.

workflow_model:
  workflow_id: <kebab-case>
  workflow_name: <human-readable title>
  version: <semver, starting 0.1.0>
  canvas_ref: <agent_id from s1-operating-context-canvas>
  discovery_protocol_ref: <e4-workflow-discovery-protocol instance or file reference>
  last_updated: <ISO-8601 date>
  status: <draft | reviewed | operational>

  scope_statement: >
    <One sentence: what this workflow starts with, what it produces, and
    what is explicitly out of scope.>

  trigger:
    type: <event | schedule | manual | api>
    source: <connector or human role>
    condition: <expression or description>

  steps:
    - step_id: <kebab-case>
      name: <short title>
      type: <action | decision | gate | handoff | stop>
      classification: <deterministic | judgment | unsafe-to-automate>
      owner: <role_id>
      inputs: [<source: field format>]
      outputs: [<destination: artifact format>]
      branches:
        - edge_id: <condition_id>
          label: <human-readable>
          branch_to: <step_id>
      hitl_gate: <true | false>
      provenance_emission: required
    # add steps as needed

  terminal_states:
    - terminal_id: <kebab-case>
      description: <what constitutes completion or termination here>
      output_type: <success | stop | escalation-pending>
```

Scaffold 2 — Decision Map

Purpose: The explicit record of every decision point in the workflow — conditions, branches, escalation triggers, and default paths — so that decision logic is readable, auditable, and testable without executing the agent.

```
# DECISION MAP
# One entry per decision-carrying step (type: decision or gate) in the workflow model.

decision_map:
  workflow_ref: <workflow_id from Scaffold 1>
  version: <semver; must match workflow_model.version>

  nodes:
    - node_id: <matches step_id in Scaffold 1>
      name: <short title>
      decision_type: <branch | gate | exception | stop>
      question: <the single question this node answers>

      conditions:
        - condition_id: <kebab-case>
          expression: <formal or pseudocode rule>
          label: <human-readable outcome label>
          confidence_required: <high | medium | any>
          branch_to: <step_id>

      default_branch:
        branch_to: <step_id>
        requires_hitl: <true | false>
        rationale: <why this is the safe default>

      escalation_triggers:
        - trigger_expression: <condition>
          threshold_ref: <s1-threshold-escalation-spec section>
          escalation_type: <hitl-gate | fallback | stop>
          target_role: <role>
          payload_summary: <brief description of what is passed>

      stop_conditions:
        - stop_id: <kebab-case>
          trigger: <condition>
          reason: <why workflow terminates>
          recovery_path: <next action>

      exception_handlers:
        - failure_mode_ref: <failure_id from s1-failure-mode-register>
          trigger: <detectable_signal>
          response: <agent action>
          branch_to: <step_id or terminal>
    # add nodes as needed
```

Scaffold 3 — Handoff Policy

Purpose: The governing policy for every ownership transfer in the workflow — what is passed, what must be accepted, and what happens when the handoff times out — so that no transition is ambiguous at runtime.

```
# HANDOFF POLICY
# One entry per handoff in the workflow (agent→human, human→agent, agent→agent).

handoff_policy:
```

```
  workflow_ref: <workflow_id from Scaffold 1>
  version: <semver>

  invented_step_prohibition: >
    No handoff may initiate a step not present in the validated step inventory
    from e4-workflow-discovery-protocol. An agent that proposes an unlisted step
    at a handoff boundary must surface it as an anomaly, not execute it.

  handoffs:
    - handoff_id: <kebab-case>
      from_owner: <role_id or "human">
      to_owner: <role_id or "human">
      trigger_step: <step_id>
      trigger_condition: <expression>

      payload:
        required_fields: [<field list>]
        confidence_bands_required: [<field: band>]
        provenance_record_included: <true | false>

      acceptance_condition: >
        <What the receiving owner must explicitly do for the handoff to be
        considered accepted. Silence is never acceptance.>

      resume_step: <step_id>
      timeout_minutes: <integer>
      timeout_action: <re-notify | escalate | stop>
      escalation_target_on_timeout: <role>

      decision_origin_on_resume: <workflow-step | human-override | escalation>
      c5_correlation_id_required: <true | false>
    # add handoffs as needed

  sla_summary:
    - role: <role>
      handoff_ids: [<list>]
      aggregate_sla_minutes: <integer>
      breach_action: <escalation action>
```

Scaffold 4 — Agent Task Spec

Purpose: The operational spec a developer uses to implement a specific agent role — its scope, tools, constraints, output format, and prohibited actions — derived from e4-agent-role-specification.

```
# AGENT TASK SPEC
# One instance per agent role in e4-agent-role-specification.

agent_task_spec:
  role_id: <matches role_id in e4-agent-role-specification>
  role_name: <short title>
  workflow_ref: <workflow_id>
  version: <semver>

  objective: >
    <One sentence: what this agent role must produce, starting from what input.>

  owned_steps: [<step_id list>]
  owned_branches: [<node_id: condition_id format>]

  system_prompt_constraints:
    - <Constraint that must appear in or be enforced by the system prompt>
    - <Constraint on output format>
    - <Hard prohibition>
```

```
  tool_registry:
    - tool_id: <connector or API>
      invocation_rule: <auto | confirm | human-approval | blocked>
      permitted_operations: [<read | append | query | etc.>]
      scope_limit: <one sentence on what is in scope>
      blocked_operations: [<list of what this role may not do with this tool>]

  context_requirements:
    - source: <connector or prior step_id>
      field: <field or schema pointer>
      confidence_band_required: <high | medium | any>
      freshness_budget_hours: <integer or null>
      on_missing: <stop | fallback | escalate>

  output_spec:
    type: <draft | recommendation | flag | routing-decision | filed-action>
    schema_pointer: <URI or filename>
    reversibility: <reversible | partially-reversible | irreversible>
    must_include: [<required fields in every output>]
    must_not_include: [<fields or content types forbidden in output>]

  hitl_obligations:
    - trigger: <condition>
      payload_required: [<fields>]
      gate_type: <hitl-gate | policy-review>
      reviewer_role: <role>

  prohibited_actions:
    - <explicit prohibition with rationale>

  failure_modes_to_monitor: [<failure_id from s1-failure-mode-register>]

  accountability_attribution: <agent | human | shared>
  post_condition_check: <expression verifying the role fulfilled its objective>
```

Scaffold 5 — Exception Register

Purpose: The enumerated catalog of exceptions the workflow may encounter — their triggers, expected responses, and recovery paths — so that edge-case handling is explicit and auditable rather than implicit in agent behavior.

```
# EXCEPTION REGISTER
# One entry per exception type in the workflow; pre-populated from
# failure_modes in e4-task-decomposition-framework and exception_handlers
# in e4-decision-logic-builder.

exception_register:
  workflow_ref: <workflow_id>
  version: <semver>

  exceptions:
    - exception_id: <kebab-case>
      name: <short title>
      failure_mode_ref: <failure_id from s1-failure-mode-register>
      affected_steps: [<step_id list>]
      severity: <critical | high | medium | low>
      detectable_signal: <observable expression; mirrors failure mode signal>
      expected_response: >
        <What the agent must do immediately on detection. Start with the
        action verb: "Cancel", "Block", "Escalate", "Quarantine", "Stop".>
      recovery_path: <sequence of recovery actions>
      hitl_required: <true | false>
      provenance_required: <true | false>
      test_seed_ref: <s2-eval-test-seed-matrix entry id or null>
```

```
      last_triggered: <ISO-8601 date or null>
      trigger_count_30d: <integer or null>

  divergence_exceptions:
    # Exceptions surfaced in discovery (e4-workflow-discovery-protocol divergence_log)
    # that represent a risk class of high or critical.
    - divergence_id: <divergence_id from discovery>
      description: <what the divergence was>
      exception_id: <exception_id above that covers it>
      resolved_in_model: <true | false>
      resolution_notes: <free text>
```

Scaffold 6 — Workflow Test Plan

Purpose: The pre-production and ongoing test plan for this workflow — test cases, expert comparison targets, and provenance audit requirements — derived from e4-operational-test-harness.

```
# WORKFLOW TEST PLAN
# One instance per workflow. Re-run after every model change.

workflow_test_plan:
  workflow_ref: <workflow_id>
  harness_ref: e4-operational-test-harness
  version: <semver>
  last_run_date: <ISO-8601 date or null>
  last_run_status: <not-run | pass | fail | partial>

  test_cases:
    - case_id: <TC-NNN format>
      type: <normal | edge | ambiguous | adversarial | regression>
      description: <one sentence>
      input_fixture_ref: <file or dataset pointer>
      expected_output: <brief description or schema pointer>
      expected_branch: <edge_id>
      expected_decision_origin: <workflow-step | escalation | fallback | human-override>
      hitl_should_fire: <true | false>
      failure_mode_tested: <failure_id or null>
      seed_source: <s2-eval-test-seed-matrix entry or "manual">
      last_result: <pass | fail | partial | not-run>
      last_result_date: <ISO-8601 date or null>
    # add cases as needed

  expert_comparison_targets:
    routing_accuracy_pct: 92
    escalation_accuracy_pct: 95
    rationale_quality_mean: 4.0
    evidence_citation_pct: 98
    hitl_gate_precision_pct: 90
    hitl_gate_recall_pct: 95
    prohibited_action_suppression_pct: 100

  provenance_audit_requirements:
    - Every executed step has a C5 record in s3-provenance-metadata-schema
    - Every HITL gate event has a matching correlation_id in the C5 record
    - Every stop condition has a C5 record
    - No C5 record contains an inlined heavy payload
    - Hash spot-check passes for a minimum of 3 randomly selected step records

  regression_suite_gate:
    minimum_cases: <integer; at least 10 for operational tier, 50 for high-stakes>
    run_on_trigger: [model-change, decision-logic-change, threshold-change, connector-
change]
    fail_action: freeze-deploy
```

```
sign_off:
  qa_reviewer: <name or role>
  domain_expert_reviewer: <name or role>
  sign_off_date: <ISO-8601 date or null>
  sign_off_notes: <free text>
```

Filling Sequence and Dependencies

Fill scaffolds in this order. Do not skip a scaffold because a step "seems obvious" — every gap in the package is a future failure mode.

1. Scaffold 1 (Workflow Model) — requires completed step inventory from e4-workflow-discovery-protocol and the classification table from e4-task-decomposition-framework.

2. Scaffold 2 (Decision Map) — requires completed decision nodes from e4-decision-logic-builder.

3. Scaffold 3 (Handoff Policy) and Scaffold 4 (Agent Task Spec) in parallel — both require e4-agent-role-specification.

4. Scaffold 5 (Exception Register) — requires the failure-mode linkage from e4-task-decomposition-framework and the divergence log from e4-workflow-discovery-protocol.

5. Scaffold 6 (Workflow Test Plan) — requires all of the above plus the agent-vs-expert targets from e4-operational-test-harness.

Once filled, the package constitutes the full deliverable. Pass it to the client, the deployment team, or store as the workflow version artifact. Version the package alongside the workflow model.