

Multi-Agent Workflow Planner

A practical planning template for designing coordinated agent workflows with clear tasks, dependencies, outcomes, and human oversight.

1. Define the Workflow

Before assigning work to multiple agents, define the workflow in plain business language. A multi-agent workflow works best when the purpose is clear, the expected result is measurable, and each participant understands where automation begins and where human judgment is still required. This section helps you describe the workflow as a business process first, not as a technical build.

1.1 Workflow / Process Name

Give the workflow a short, specific name that tells people what the workflow does. Avoid vague labels such as “AI Automation” or “Agent Workflow.” A strong name should identify the business process, the trigger, or the output.

- **Weak name:** AI Support Agent
- **Better name:** IT Incident Triage and Response Workflow
- **Strong name:** Finance ERP Login Incident Triage Workflow

Example: If the workflow is designed to support service-desk teams, the process name could be “Priority Incident Classification and Escalation Planner.” This name immediately shows that the workflow classifies incidents and supports escalation decisions.

1.2 What Are You Trying to Accomplish?

Describe the business problem the workflow is meant to solve. Focus on the reason for the workflow: reducing manual effort, improving consistency, speeding up response time, increasing accuracy, or coordinating multiple specialized tasks. A useful description should explain the current pain point and why a multi-agent approach is appropriate.

- What process is slow, repetitive, or inconsistent today?
- What decisions or outputs must be improved?
- Why is one agent not enough?
- Which parts require specialist knowledge, different tools, parallel processing, or review?

Example: The workflow may aim to triage incoming IT incidents, extract the affected service, classify severity, search known errors, draft a first response, and recommend escalation. One agent could struggle to perform all these tasks reliably, while specialized agents can divide the work into intake analysis, knowledge search, customer communication, and escalation recommendation.

1.3 What Is the Desired Outcome?

Define the final output that the workflow should produce. The outcome should be observable and useful to the people who depend on the workflow. In multi-agent

design, the final outcome often combines several intermediate outputs created by different agents.

- A completed decision or recommendation
- A drafted communication ready for review
- A structured report or summary
- An updated ticket, file, record, or workflow status
- A clear handoff to a human approver or downstream team

Example: For a procurement workflow, the desired outcome could be a supplier comparison summary with risk flags, pricing observations, missing-document alerts, and a recommended shortlist for human approval.

1.4 Who Will Use the Workflow?

Identify the users, reviewers, approvers, and stakeholders who will interact with the workflow. This helps determine the level of detail, controls, permissions, and explanation required. A workflow for frontline staff may need simple prompts and suggested actions, while a workflow for managers may need dashboards, exception summaries, and audit trails.

- **Primary users:** People who run or trigger the workflow.
- **Reviewers:** People who check the workflow output before action is taken.

- **Approvers:** People who authorize sensitive actions or final decisions.
- **Impacted teams:** Teams affected by the workflow outcome.
- **Administrators:** People responsible for configuration, monitoring, and improvement.

Example: In an HR screening workflow, recruiters may trigger the workflow, hiring managers may review summaries, compliance teams may validate fairness controls, and HR operations may monitor completion status.

2. Break the Workflow Into Tasks

Once the workflow is defined, break it into smaller tasks. Multi-agent systems are most effective when each agent has a clear responsibility, a defined input, a specific output, and boundaries around what it should not do. Task decomposition also makes it easier to identify which steps can happen in parallel, which steps require sequencing, and where human input must remain part of the process.

2.1 List Each Step Involved

Start by writing the workflow as a sequence of steps from trigger to final output. At this stage, do not worry about which agent will perform each task. The goal is to capture the work clearly and completely.

- Trigger: What starts the workflow?
- Input collection: What information must be gathered?
- Analysis: What must be interpreted, classified, or compared?
- Action planning: What recommendations or decisions are created?
- Output creation: What document, message, update, or record is produced?
- Review and approval: What must a human validate before execution?
- Completion: How is the workflow closed, logged, or handed off?

Example task list for an incident workflow:

1. Receive incident details from the service desk ticket.
2. Extract affected application, user group, location, symptoms, and time of impact.
3. Classify severity and urgency.
4. Search known errors, recent changes, and similar past incidents.
5. Draft a first response to the requester.
6. Recommend an escalation path or resolver group.
7. Ask a human analyst to approve the final response or escalation.
8. Update the ticket with the approved summary and next action.

2.2 Identify Tasks That Can Run Independently

Independent tasks are steps that do not need the result of another step before they can begin. Identifying these tasks is important because they can often run in parallel, reducing total workflow time. In a multi-agent workflow, parallel execution is useful when different agents can work from the same initial input and produce separate outputs for later synthesis.

- A research agent can gather background information while a classification agent analyzes the ticket.

- A compliance agent can check policy constraints while a drafting agent prepares a response.
- A data agent can retrieve metrics while a reporting agent prepares the report structure.

Example: In a supplier onboarding workflow, one agent can verify submitted documents, another can check sanctions or compliance requirements, and another can summarize supplier profile information. These tasks can run independently because they use the same initial supplier package but do not depend on one another's outputs.

2.3 Identify Tasks That Depend on Previous Steps

Dependent tasks require an earlier result before they can begin. These dependencies define the sequence of the workflow and help the orchestrator decide what to run first, what to wait for, and what to do if a previous step fails. Clear dependency mapping prevents agents from acting on incomplete or unverified information.

- Severity classification may depend on extracting affected users and business impact.
- A customer response may depend on the approved incident status and next action.
- A final recommendation may depend on comparing multiple agent outputs.
- An automated system update may depend on human approval.

Example: In a loan application workflow, an eligibility recommendation should not be generated until income documents, credit summary, policy checks, and risk indicators have been reviewed. The recommendation agent depends on outputs from document extraction, risk assessment, and compliance validation agents.

2.4 Note Where Human Input Is Currently Required

Human input should be identified deliberately, not added as an afterthought. Some steps may require human review because they involve judgment, risk, compliance, customer sensitivity, exceptions, or irreversible action. In these areas, agents can prepare recommendations, evidence, and draft outputs, but people should approve or override the action.

- **Approval points:** Where must a person approve before the workflow continues?
- **Exception handling:** When should the workflow pause and ask for guidance?
- **Policy-sensitive steps:** Which actions require compliance, privacy, or security review?
- **Customer-facing communication:** Which messages need human confirmation before sending?
- **High-impact decisions:** Which outcomes affect money, access, employment, legal matters, or critical operations?

Example: In an ERP access outage workflow, agents may classify the incident, identify affected users, draft the service update, and recommend escalation. However, a human incident manager should approve severity level, external communication, and any major incident declaration before action is taken.

3. Decide: Single Agent or Multiple Agents?

After breaking the workflow into tasks, decide whether the work should be handled by one capable agent or several specialized agents. A single-agent design is usually simpler, easier to test, and easier to govern. A multi-agent design may be useful when the workflow crosses domains, requires different tools, includes parallel work, or needs independent review. The goal is not to add more agents because it sounds advanced; the goal is to choose the lowest-complexity design that can reliably deliver the desired outcome.

3.1 Can the Work Be Split Into Separate Tasks?

Use multiple agents only when the workflow naturally separates into meaningful work packages. If every step depends tightly on the previous step and requires the same reasoning context, one agent may be more efficient. If the workflow contains clearly different streams of work, such as research, classification, drafting, validation, and reporting, separate agents may improve quality and speed.

- **Good fit for a single agent:** A short, linear task such as summarizing one support ticket and drafting one response.
- **Good fit for multiple agents:** A workflow where one agent extracts facts, another checks policy, another searches knowledge articles, and another drafts the final response.

- **Warning sign:** If agents would mostly repeat the same work or wait for each other constantly, the workflow may not need multiple agents.

Example: A project status update workflow can often be split into separate tasks: one agent gathers schedule data, another reviews risks, another summarizes budget health, and another prepares the executive update. These streams can later be combined into one final report.

3.2 Does Each Task Need Different Skills or Tools?

Different agents are most useful when different tasks require different skills, data sources, systems, or decision rules. If one task needs document analysis, another needs database lookup, another needs policy review, and another needs communication drafting, specialization can make the workflow clearer and safer. Each agent can be given only the tools and access it needs, which also supports better control.

- **Research agent:** Searches internal knowledge bases, policies, or approved external sources.
- **Data agent:** Retrieves structured data from systems such as CRM, ERP, ticketing, or reporting tools.
- **Compliance agent:** Checks whether the proposed action follows policy, legal, privacy, or security rules.
- **Drafting agent:** Converts analysis into a clear email, ticket update, report, or customer message.

- **Review agent:** Looks for gaps, contradictions, missing evidence, or unsupported conclusions.

Example: In a finance reconciliation workflow, a data extraction agent may read transaction files, a matching agent may compare ledger entries, an exception agent may identify breaks, and a reporting agent may prepare the daily reconciliation summary for review.

3.3 Is the Added Cost Justified?

Multiple agents add coordination effort, testing needs, monitoring complexity, and cost. The decision should be based on business value, not design preference. If multiple agents reduce cycle time, improve accuracy, lower operational risk, or support scale, the added cost may be justified. If the workflow is simple and low risk, a single agent may deliver the same value with less overhead.

- **Cost factors:** More prompts, more model calls, more orchestration logic, more logs, and more failures to troubleshoot.
- **Benefit factors:** Faster completion, better specialization, stronger review, clearer accountability, and easier scaling.
- **Governance factors:** More agents may require more access controls, audit trails, testing, and ownership definitions.

Example: A customer support FAQ response may not justify a multi-agent design. However, a regulatory complaint workflow may justify multiple agents because it may

require case analysis, policy review, evidence retrieval, response drafting, and human approval.

3.4 Would a Single Agent Be Enough?

Before building multiple agents, test whether a single well-instructed agent can complete the workflow safely and consistently. A single agent may be enough when the task is narrow, the data sources are limited, the output format is simple, and the risk level is low. Starting simple also helps teams learn what actually needs specialization before adding orchestration.

- Use a single agent if the workflow is short, repeatable, and easy to validate.
- Use a single agent if all steps use the same tools and the same context.
- Use multiple agents if tasks can run in parallel or require different access rights.
- Use multiple agents if independent review or separation of duties is important.

Decision rule: Start with the simplest design that works. Move from single agent to multiple agents only when there is clear evidence that specialization, parallel work, separation of duties, or scale will improve the outcome.

4. Define Your AI Agents

Once you decide to use multiple agents, define each agent clearly. A useful agent definition should explain what the agent is called, what role it plays, what it is responsible for, what instructions guide its behavior, which tools it can access, and what it must not do. Without clear definitions, agents may overlap, produce inconsistent results, or take actions outside their intended scope.

4.1 Agent Name

Give each agent a clear name that reflects its function. The name should help users and administrators understand the agent's purpose without reading a long description.

Avoid generic names such as "Agent 1" or "Helper Bot."

- **Good names:** Incident Intake Agent, Policy Review Agent, Data Retrieval Agent, Response Drafting Agent.
- **Poor names:** AI Agent, Main Bot, Assistant 2, Smart Processor.
- **Naming tip:** Combine the work area with the agent's function, such as "Procurement Risk Review Agent."

4.2 Agent Role

The agent role explains why the agent exists in the workflow. It should describe the function the agent performs, not just the task it completes. A role can be similar to a job title in a human team: analyst, reviewer, researcher, coordinator, validator, or drafter.

- **Analyst role:** Interprets inputs, identifies patterns, and produces findings.
- **Reviewer role:** Checks quality, completeness, and consistency.
- **Coordinator role:** Routes work, combines outputs, and tracks workflow status.
- **Drafter role:** Converts structured findings into business-ready communication.

Example: In an ITSM workflow, the “Incident Classification Agent” role is to analyze the ticket details and determine category, urgency, impact, and likely severity. It should not draft customer messages unless that is part of its assigned role.

4.3 Main Responsibility

The main responsibility is the specific output the agent is accountable for producing. Keep this focused. If one agent has too many responsibilities, it becomes harder to test, govern, and troubleshoot. A good responsibility statement includes the input the agent receives, the work it performs, and the output it returns.

- **Input:** Incident description, user impact, affected system, screenshots, logs, or prior ticket history.

- **Work performed:** Extract key facts, classify the issue, compare against known patterns, and flag missing data.
- **Output:** A structured triage summary with category, urgency, impact, confidence level, and open questions.

Example responsibility statement: “The Policy Review Agent reviews proposed workflow actions against approved policy rules and returns a pass, warning, or escalation recommendation with reasons.”

4.4 Instructions

Instructions tell the agent how to behave, what steps to follow, what format to produce, and when to stop or escalate. Strong instructions reduce ambiguity and make outputs more predictable. They should be specific enough to guide the agent, but not so rigid that the agent cannot handle reasonable variation.

- State the goal of the agent.
- List the steps the agent should follow.
- Define the expected output format.
- Set quality checks before the output is returned.
- Tell the agent when to ask for missing information.
- Tell the agent when to escalate to a human.

Example instruction: “Review the incident ticket, extract the affected service, classify urgency and impact using the severity matrix, identify missing information, and return a structured summary. If business impact is unclear, ask for clarification instead of guessing.”

4.5 Tools and Access Required

Each agent should have access only to the tools, systems, and data required to complete its responsibility. This supports security, privacy, and operational control. Tool access should be mapped to the agent’s role rather than granted broadly to every agent in the workflow.

- **Knowledge access:** Policies, standard operating procedures, knowledge articles, FAQs, and training documents.
- **System access:** Ticketing tools, CRM, ERP, HRIS, project management platforms, or reporting systems.
- **Communication access:** Email drafts, Teams messages, notification templates, or approval requests.
- **Data access:** Logs, records, reports, dashboards, and structured datasets.
- **Control limits:** Read-only access, draft-only permissions, approval-required actions, or restricted fields.

Example: A Response Drafting Agent may need access to approved communication templates and ticket context, but it should not need permission to close tickets or change severity levels. Those rights should remain with a human analyst or a controlled workflow step.

4.6 What the Agent Should Not Do

Defining limits is as important as defining responsibilities. Agents should be told what actions are out of scope, which decisions require human approval, and which assumptions they must not make. This reduces risk and prevents agents from overstepping their intended role.

- Do not make final approval decisions for high-risk actions.
- Do not access systems or data outside assigned permissions.
- Do not send customer-facing messages without required approval.
- Do not invent missing facts, dates, policy rules, or evidence.
- Do not override human decisions or documented business rules.
- Do not continue if required information is missing; ask for clarification or escalate.

Example guardrail: “The Escalation Recommendation Agent may suggest the most appropriate resolver group, but it must not reassign the ticket automatically unless the reassignment has been approved by a human or an approved routing rule.”

5. Map Agent Hand-Offs

Agent hand-offs describe how work moves from one agent to another. A hand-off is more than a message; it is a controlled transfer of task state, evidence, assumptions, decisions, and open questions. Good hand-offs reduce rework, prevent context loss, and make the workflow easier to audit. Poor hand-offs create confusion because the next agent may not know what was done, why it was done, or what still needs attention.

5.1 Which Agent Starts the Process?

Identify the first agent in the workflow and define what triggers it. The starting agent is often responsible for intake, validation, and routing. It receives the initial request, checks whether enough information is available, and decides which downstream agents should be activated.

- **Trigger source:** A ticket, email, form submission, chat message, scheduled job, API event, or manual user request.
- **Starting agent:** The agent that receives the trigger and prepares the initial task package.
- **Routing logic:** The rules used to decide which specialist agents should work next.
- **Validation check:** The minimum information required before the workflow can continue.

Example: In an IT incident workflow, the Incident Intake Agent may start the process when a new high-priority ticket is created. It extracts the affected system, user impact, error message, location, and time of occurrence before handing the case to classification, knowledge search, and response drafting agents.

5.2 What Does Each Agent Send to the Next?

Each agent should send a structured output, not a vague narrative. This makes the next step easier to automate and review. A good hand-off package should include the agent's conclusion, supporting evidence, confidence level, unresolved questions, and any warning flags.

- **Summary:** What the agent found or completed.
- **Key fields:** Structured data such as category, priority, customer, system, amount, region, or date.
- **Evidence:** Source documents, logs, policy references, screenshots, or system records used.
- **Decision or recommendation:** The proposed next action.
- **Confidence level:** High, medium, or low confidence with a short reason.
- **Open items:** Missing information, unresolved risks, or questions for the next agent or a human reviewer.

Example: A Policy Review Agent may send the Response Drafting Agent a simple package: “Policy status: warning; reason: customer message references an unconfirmed restoration time; required change: remove exact ETA unless confirmed by incident manager; confidence: high.”

5.3 What Information Needs to Be Passed?

Only pass information that the next agent needs to complete its job. Passing too little context causes errors; passing too much context creates noise, increases cost, and may expose unnecessary data. The goal is role-scoped context: each agent receives the information required for its responsibility, while sensitive or irrelevant information is excluded.

- **Task objective:** What the next agent is expected to do.
- **Required inputs:** Facts, records, IDs, documents, links, or previous outputs needed for the task.
- **Constraints:** Policies, format requirements, deadlines, approval rules, or risk limits.
- **Decision history:** Important choices already made and why.
- **Traceability:** Where the information came from and whether it has been verified.

Example: A Finance Exception Agent does not need the full customer profile if its job is only to investigate unmatched payments. It may need transaction ID, account code, amount, date, payment reference, ledger entry, and reconciliation rule. Customer-sensitive details should be excluded unless needed.

5.4 Where Could Information Get Lost?

Information can get lost when outputs are unstructured, assumptions are not documented, source references are removed, or hand-offs happen through free-text messages only. In multi-agent workflows, even small omissions can lead to incorrect downstream actions. Identify likely failure points and design controls to reduce context loss.

- **Missing source links:** The next agent cannot verify where a fact came from.
- **Unclear assumptions:** An agent treats a guess as confirmed information.
- **Format mismatch:** One agent sends free-text output when the next agent expects structured fields.
- **Over-summarization:** Important details are removed during compression or summarization.
- **Permission gaps:** The next agent cannot access the necessary system or document.

- **State drift:** A system record changes after an agent has already used an older value.

Control tip: Use a standard hand-off template for every agent. Include task ID, source links, required fields, decision status, confidence level, timestamp, and open questions.

This gives the workflow a traceable record of what happened at each stage.

6. Map Tools & Data Access

Tool and data mapping defines what each agent can read, retrieve, create, update, or request. This step turns the workflow design into an operational access model. It also helps prevent over-permissioning, data leakage, and uncontrolled actions. Use the principle of least privilege: each agent should receive only the access needed to complete its assigned responsibility.

6.1 CRM or Business Systems

Many workflows need information from CRM, ERP, HRIS, ITSM, procurement, finance, or project management systems. Map exactly which system each agent needs and whether the agent requires read-only access, draft access, update access, or approval-based action access.

- **CRM:** Customer profile, account status, opportunity details, service history, and contact preferences.
- **ERP:** Purchase orders, invoices, inventory, finance records, vendor master data, or access status.
- **ITSM:** Tickets, incidents, problems, changes, service catalog items, and knowledge articles.
- **HRIS:** Employee records, role data, training status, onboarding tasks, and approval chains.

- **Project tools:** Tasks, milestones, dependencies, risks, issues, and status updates.

Example: A Customer Renewal Agent may need read access to CRM account data, contract dates, usage records, and support history. It may draft a renewal summary, but it should not change contract values or send renewal offers without approval.

6.2 Documents and Knowledge Bases

Documents and knowledge bases provide agents with approved reference material.

These may include standard operating procedures, product manuals, policies, training guides, previous incident reports, architectural diagrams, FAQs, and checklist templates.

Define which sources are authoritative so agents do not rely on outdated or unofficial information.

- **Approved sources:** Current policies, SOPs, knowledge articles, product documentation, and governance standards.
- **Source priority:** Which source wins when two documents conflict.
- **Freshness check:** How the agent detects outdated content or superseded documents.
- **Citation requirement:** Whether the agent must include the source used in its output.
- **Restricted content:** Sensitive documents that require role-based access or human approval.

Example: A Policy Review Agent should use the current approved policy repository, not an old downloaded copy of a policy. If the policy version cannot be verified, the agent should flag the issue and ask for confirmation.

6.3 Databases and APIs

Some agents need structured data from databases or APIs. Define the query patterns, allowed fields, rate limits, error handling rules, and update permissions. Agents should not be allowed to run unrestricted queries or perform write actions unless there is a clear control model.

- **Read queries:** What data can the agent retrieve?
- **Write actions:** Can the agent create, update, or delete records?
- **Rate limits:** How many calls can the agent make in a given period?
- **Data validation:** What checks happen before using or updating data?
- **Error handling:** What should the agent do when an API fails or returns incomplete data?

Example: A Reporting Agent may call a project management API to retrieve milestones, overdue tasks, and risk status. It may use that data to draft a weekly status report, but updates to project dates or risk ratings should require human approval.

6.4 MCP Connections

Model Context Protocol connections can provide a standard way for agents to access external tools and contextual data. MCP can be useful when multiple agents need consistent access to tools such as document repositories, databases, service platforms, developer tools, or enterprise applications. Treat each MCP connection as a governed integration, not as an open doorway to every tool.

- **Connection purpose:** What tool, system, or dataset does the MCP connection expose?
- **Authentication:** Which credentials, tokens, or identity model are used?
- **Permitted actions:** What operations can the agent perform through the connection?
- **Logging:** What requests and responses are recorded for audit purposes?
- **Provider trust:** Whether the MCP server is internal, vendor-hosted, or third-party.

Example: A Knowledge Search Agent might use an MCP connection to query an approved internal knowledge base. The connection should allow search and retrieval, but not deletion, bulk export, or unrestricted access to confidential repositories.

6.5 Access Permissions

Access permissions define what each agent is allowed to see and do. Permissions should be assigned based on role, task, risk, and data sensitivity. This is especially important when agents interact with customer records, employee data, financial information, source code, security logs, or regulated content.

- **Read-only:** The agent can view information but cannot change records.
- **Draft-only:** The agent can prepare updates, messages, or records for human review.
- **Approval-required:** The agent can recommend actions, but execution requires a person or approval workflow.
- **Limited write:** The agent can update specific fields under defined business rules.
- **No access:** The agent is blocked from sensitive systems or data not needed for its job.

Example permission model: An Incident Classification Agent may have read-only access to tickets and knowledge articles. A Response Drafting Agent may create draft updates but cannot send them. A Human Approver or controlled workflow step must approve severity changes, customer notifications, or ticket closure.

7. Add Human Approval Points

Human approval points define where the workflow must pause for review before continuing. These checkpoints are especially important when agents create outputs that affect customers, money, employee records, system access, public communication, compliance obligations, or production systems. A good approval point is specific: it tells the human exactly what they are reviewing, why the review is needed, and what happens after approval, rejection, or escalation.

7.1 Where Should a Human Review the Output?

Human review should be placed at points where judgment, accountability, or risk assessment matters. The goal is not to review every small action, because that slows the workflow and reduces automation value. Instead, place review checkpoints before important decisions, external messages, high-risk updates, or irreversible actions.

- Before customer-facing communication is sent.
- Before a ticket, case, claim, or incident is closed.
- Before severity, priority, risk rating, or approval status is changed.
- Before an agent executes a write action in a business system.
- Before a workflow output is used for compliance, finance, legal, HR, or security decisions.

Example: In a major incident workflow, the agent may draft a stakeholder update, but a human incident manager should review the message before it is sent. The reviewer should confirm the impact, current status, approved workaround, and any expected restoration timeline.

7.2 Which Actions Require Approval?

Actions require approval when they are high-impact, difficult to reverse, externally visible, or subject to policy control. Approval requirements should be based on the action type and risk level, not only on the agent's confidence. Even a confident agent should not take sensitive actions without required authorization.

- **External communication:** Sending emails, customer notices, public updates, or regulatory responses.
- **Financial action:** Approving payments, refunds, credits, invoices, or purchase orders.
- **Access action:** Granting, revoking, or changing user permissions.
- **System action:** Closing incidents, deploying changes, modifying production data, or restarting services.
- **People-related action:** Updating HR records, screening outcomes, disciplinary notes, or performance-sensitive information.

Example: A Procurement Risk Agent may recommend rejecting a supplier because of missing documents or risk flags, but the actual supplier rejection decision should require a procurement owner or compliance reviewer to approve the recommendation.

7.3 What Happens If an Agent Produces an Incorrect Result?

Incorrect outputs should be handled through a defined correction path. The workflow should not simply continue after a mistake is found. It should pause, record the issue, identify the responsible step, and route the work for correction or human review. This prevents the same error from spreading through downstream agents.

- Stop the workflow if the error affects a high-risk action or decision.
- Record the incorrect output and the source information used.
- Identify whether the issue came from missing data, poor instructions, wrong tool access, or weak validation.
- Send the task back to the relevant agent with corrected context.
- Escalate to a human if the agent cannot correct the issue confidently.

Example: If an Incident Classification Agent marks a payroll outage as low priority even though many users are affected, the workflow should stop the automated path, route the case to a human incident manager, and update the agent's classification rules or examples before reuse.

7.4 What Happens When a Task Needs Escalation?

Escalation rules tell the workflow when agent handling is no longer enough. Escalation may be needed because of missing information, conflicting evidence, low confidence, policy uncertainty, repeated failure, user dissatisfaction, or high business impact. Every escalation should include a concise summary, evidence, attempted actions, and the decision needed from the human or specialist team.

- **Functional escalation:** Route to a specialist team when the issue requires domain expertise.
- **Management escalation:** Route to a manager when the decision affects cost, customers, or deadlines.
- **Security escalation:** Route to security when access, sensitive data, or suspicious activity is involved.
- **Compliance escalation:** Route to legal, risk, or compliance when policy interpretation is uncertain.
- **Technical escalation:** Route to engineering, platform, or system owners when root cause analysis is required.

Example escalation package: “Issue: ERP login failures affecting payroll users. Impact: payroll processing delayed. Agent actions completed: extracted ticket details, checked knowledge base, found no current outage notice. Confidence: medium. Escalation requested: ERP support team to validate authentication service status.”

8. Add Guardrails

Guardrails are the operating limits that keep the workflow safe, controlled, and auditable. They should be enforced through workflow design, permissions, validation rules, and monitoring—not only through agent instructions. A guardrail prevents an agent from exceeding its role even if the prompt is incomplete, the user asks for something risky, or a tool returns unexpected data.

8.1 Maximum Number of Loops

Agents sometimes repeat steps when they are uncertain, when a tool fails, or when they keep trying to improve an answer. Set a maximum number of loops so the workflow does not run indefinitely or consume unnecessary resources. When the loop limit is reached, the workflow should stop and escalate with a summary of what was attempted.

- Set a retry limit for failed tool calls.
- Set a refinement limit for repeated drafting or review cycles.
- Stop the workflow after repeated low-confidence outputs.
- Escalate when the same agent fails the same task more than the allowed limit.

Example: A Knowledge Search Agent may be allowed three search attempts. If it cannot find a relevant approved article after three attempts, it should stop and ask a human analyst to confirm whether the knowledge base is missing content.

8.2 Permission Limits

Permission limits restrict what each agent can access or change. These limits should be applied at the system level wherever possible. An agent should not be trusted with broad permissions just because its instructions say it will only use them responsibly.

- Use read-only access unless the agent truly needs write access.
- Use draft-only access for communication and record updates.
- Restrict sensitive fields such as compensation, health, credentials, and financial account details.
- Require approval before any create, update, delete, send, deploy, or close action.

8.3 Data Restrictions

Data restrictions define what information agents can use, store, summarize, or pass to other agents. This is important for privacy, confidentiality, regulatory compliance, and internal security. The workflow should avoid sending unnecessary personal, financial, health, credential, or confidential business data to agents that do not need it.

- Mask or remove sensitive fields when they are not required.
- Pass only role-relevant context to each agent.
- Prevent agents from exporting large datasets without approval.
- Define retention rules for logs, summaries, and intermediate outputs.

- Block use of outdated, unofficial, or unapproved data sources.

8.4 Error-Handling Rules

Error-handling rules explain what the workflow should do when something goes wrong. Failures may come from unavailable tools, missing records, conflicting information, invalid outputs, low confidence, or unexpected user requests. The workflow should fail safely, meaning it should stop or escalate rather than continue with uncertain information.

- If a tool fails, retry only within the allowed retry limit.
- If required data is missing, ask for clarification or escalate.
- If two sources conflict, use the approved source priority rule.
- If confidence is low, pause before producing final output.
- If an output fails validation, return it to the responsible agent for correction.

8.5 Escalation Rules

Escalation rules define when the workflow must involve a person or a specialist team. These rules should be visible to users and administrators so they understand why the workflow paused. Escalation should provide evidence, not just a notification.

- Escalate when the task affects safety, security, compliance, finance, customers, or employee records.

- Escalate when the agent confidence falls below the defined threshold.
- Escalate after repeated tool failures or repeated correction loops.
- Escalate when the agent detects conflicting evidence or unclear policy interpretation.
- Escalate when the user requests an action outside the workflow scope.

8.6 Logging and Monitoring

Logging and monitoring provide the evidence needed to understand what the workflow did, why it did it, and who approved it. Logs should capture actions, inputs, agent outputs, tool calls, approvals, errors, and final outcomes. Monitoring should highlight unusual behavior, repeated failures, access attempts, long-running tasks, and high-risk actions.

- Track which agent performed each step.
- Record source documents, tool outputs, and decision evidence.
- Log approvals, rejections, edits, and escalation outcomes.
- Monitor loop counts, error rates, confidence levels, and processing time.
- Review logs regularly to improve prompts, tools, permissions, and guardrails.

9. Test the Workflow

Testing confirms whether the workflow works as designed before it is used in real operations. Testing should include normal scenarios, edge cases, missing data, tool failures, low-confidence outputs, policy-sensitive decisions, and escalation paths. A workflow is not ready just because it works once; it should work consistently across realistic scenarios.

9.1 Test Scenario

A test scenario describes the situation the workflow must handle. It should include the input, context, expected business rules, and any constraints. Create several scenarios, including easy cases and difficult cases, so you can see how agents behave under different conditions.

- Normal case: All required information is available and the workflow should complete smoothly.
- Missing-data case: Key information is absent and the agent should ask for clarification.
- Conflict case: Two sources disagree and the workflow should apply source priority rules.
- High-risk case: The workflow identifies an action that requires human approval.

- Failure case: A tool or API is unavailable and the workflow should retry, stop, or escalate.

9.2 Expected Result

The expected result defines what should happen if the workflow performs correctly. It should be specific enough to compare against the actual result. Include the expected agent outputs, hand-offs, approvals, final output, and audit record.

- The correct agents are triggered in the right order.
- Each agent produces the expected structured output.
- Required evidence and source references are preserved.
- Human approval is requested at the right checkpoint.
- The final output is complete, accurate, and ready for use.

9.3 Actual Result

The actual result records what happened during the test. Capture the full workflow path, including agent outputs, tool responses, retries, approvals, errors, and final output. Avoid rewriting the result to make it look cleaner; testing is useful because it reveals where the workflow behaves unexpectedly.

- Which agents ran?
- What did each agent produce?

- Were any outputs missing, vague, or unsupported?
- Did the workflow pause for approval at the correct point?
- Did any tool fail, return incomplete data, or create delay?

9.4 What Went Wrong?

When a test fails, identify the root cause before adjusting the workflow. Do not assume the language model is the only problem. Many workflow failures come from unclear instructions, missing data, poor access design, weak hand-off formats, tool errors, or missing guardrails.

- **Instruction issue:** The agent was not told exactly what to do or how to format the output.
- **Data issue:** The agent did not receive the information needed to complete the task.
- **Tool issue:** The required tool was unavailable, misconfigured, or returned incomplete results.
- **Access issue:** The agent lacked permission to retrieve necessary information.
- **Validation issue:** The workflow did not check whether the output was complete or correct.

9.5 Which Agent Needs Improvement?

Identify the agent or workflow component that needs improvement. Be specific about whether the problem belongs to the agent instructions, tool access, output format, validation rule, hand-off package, or approval checkpoint. Improvement should be targeted; changing every agent at once makes it difficult to know what fixed the issue.

- Update the agent's instructions with clearer steps and examples.
- Add required output fields so downstream agents receive complete information.
- Improve tool access or change the source priority rule.
- Add validation checks before the agent output is accepted.
- Add escalation rules when confidence is low or evidence is missing.

Example: If the Response Drafting Agent creates a customer message that includes an unapproved restoration time, the response agent may need stricter instructions, and the workflow may need a policy review checkpoint before messages are sent.

10. Create the Final Workflow Map

The final workflow map brings the design together in one clear view. It shows the input, agent sequence, hand-offs, human review point, and final output. This map can be used by business owners, automation builders, reviewers, risk teams, and users to understand how the workflow operates from start to finish.

10.1 Input

The input is the information that starts the workflow. It may come from a form, ticket, email, chat, API event, uploaded file, scheduled report, or manual user request. The input definition should include the required fields, optional fields, accepted formats, and validation checks.

- Trigger source: new ticket, form submission, email, or system event.
- Required fields: requester, business context, issue description, priority indicators, and supporting evidence.
- Validation: check for missing fields, duplicate requests, unsupported scope, or sensitive data.

10.2 Agent 1

Agent 1 is usually the intake, triage, or coordinator agent. Its job is to understand the input, extract key facts, check whether required data is present, and route work to the appropriate specialist agents.

- **Example name:** Intake and Routing Agent.
- **Main output:** Structured task package with key facts, missing fields, risk indicators, and downstream routing decision.
- **Hand-off:** Sends validated context to Agent 2 and, if needed, parallel tasks to other agents.

10.3 Agent 2

Agent 2 is often the analysis, research, or validation agent. It uses approved tools, systems, or knowledge sources to investigate the task and produce evidence-based findings.

- **Example name:** Research and Validation Agent.
- **Main output:** Findings, evidence, source references, confidence level, and risks.
- **Hand-off:** Sends findings to Agent 3 for drafting, recommendation, or synthesis.

10.4 Agent 3

Agent 3 usually prepares the business-ready output. It may draft a response, create a summary, generate a recommendation, assemble a report, or prepare an update for review. This agent should not introduce new unsupported claims; it should use the evidence provided by earlier agents.

- **Example name:** Output Drafting Agent.
- **Main output:** Draft communication, recommendation summary, workflow update, or final report.
- **Hand-off:** Sends the output, source evidence, confidence rating, and open questions to the human reviewer.

10.5 Human Review

The human reviewer checks the final output before action is taken. The reviewer confirms the facts, approves or rejects the recommendation, edits the message if needed, and decides whether escalation is required. Human review should be simple enough to complete quickly but detailed enough to support accountability.

- Review the proposed action and supporting evidence.
- Approve, reject, edit, or escalate the output.
- Record reviewer notes and approval decision.

- Confirm that the final action follows policy, security, and business rules.

10.6 Final Output

The final output is the approved result of the workflow. It should be clear, complete, traceable, and ready for use. Depending on the workflow, the final output may be a ticket update, email draft, completed report, recommendation, decision memo, escalation package, workflow status update, or system record.

- **For ITSM:** Approved incident summary, escalation path, and customer update.
- **For procurement:** Supplier comparison, risk flags, and approval recommendation.
- **For HR:** Candidate screening summary, missing information, and reviewer notes.
- **For finance:** Reconciliation exception summary and next action recommendation.
- **For project management:** Executive status update with risks, dependencies, and decisions needed.

Conclusion

A strong multi-agent workflow is not just a chain of AI tasks. It is a designed operating model with clear responsibilities, controlled hand-offs, role-based tool access, human approval points, guardrails, testing, and monitoring. The most effective workflow is usually the simplest design that can safely and consistently deliver the desired outcome. Start with the business process, define the agents only where specialization adds value, and keep people involved where judgment, risk, approval, or accountability is required.

Use this planner as a repeatable template: define the workflow, break it into tasks, decide the agent model, map hand-offs and access, add approval checkpoints, apply guardrails, test realistic scenarios, and document the final workflow map. When maintained properly, this approach helps teams build agentic workflows that are practical, auditable, scalable, and safe for enterprise use.

AGENTIC AI FOUNDATION CERTIFICATION

AGENTIC AI FOUNDATION, BASED ON
THE PRINCIPLES OF ETHICS AND
RESPONSIBILITY, DRIVES AI
INNOVATION.



ABOUT GSDC CERTIFICATION



LIFETIME VALIDITY

GSDC Certification is an globally
accredited certification with lifetime
validity.



EBOOK

Extensive and exclusive Ebook created by
world's experts to help you with
understanding core concepts.



CREATED BY EXPERTS

GSDC certifications are created and
authored by world's leading experts in the
field.



LEARNING MATERIALS

Get access to learning materials such as videos,
ebooks, templates, and practice exams, which
will help you clear the certification exam.

LEARNING OBJECTIVE

- Access ready-to-implement templates for agentic AI solutions.
- Develop a deep understanding of agentic AI principles.
- Prepare for real-world challenges with agentic AI applications.

Enroll now with the
code **LEARN20** To
avail **20%** discount

Enroll Now



www.gsdcouncil.org