



Agentic AI Controls, Governance and Assurance — A Reference Catalogue

How do you get agents to do what you want?

A practical guide to the controls, governance and assurance mechanisms for AI agents

Richard Dickerson · Cognostrix

VERSION 1.0 · BASELINE

September 2026

Living reference guide

113-control baseline • designed for human and machine-assisted use • intended to evolve through practitioner review, real-system dogfooding and changes in technology, threat patterns and external requirements

Central question

What combination of mechanisms gives us reasonable confidence that an agent will behave as intended — and let us detect, constrain and recover when it does not?

Status: For information. Not a standard, certification scheme, legal opinion or completeness claim.

Executive Summary

AI agents are useful precisely because they can interpret goals, reason about context, choose methods and act with a degree of autonomy. Those same properties make control different from traditional deterministic software. An agent can satisfy the literal instruction while missing the intended outcome; behave differently in a new context; follow an untrusted instruction; choose an unexpected tool path; or make a locally sensible decision that becomes harmful when combined with permissions, persistence or other agents.

The practical answer is not a single prompt, guardrail or governance framework. Effective agent control is a system of complementary mechanisms applied across the lifecycle: purpose, specification, architecture, context, authority, runtime execution, verification, observation, intervention, change and accountability. Some mechanisms influence model behaviour. Others enforce hard limits outside the model. Still others provide evidence that the system is working as intended.

This guide is a catalogue and decision aid for those mechanisms. This working guide contains 113 catalogued controls grouped into 16 families.

Each control has a stable ID, practical implementation guidance, test and evidence ideas, limitations, common complements and metadata that can be used by people or machine-assisted governance processes. The 113 controls are a baseline, not a claim of completeness.

Scope and limitations

This guide provides practical input to help teams think about, select, implement, test and evidence agent controls. It does not establish that any particular system is safe, secure, compliant or certified. It is not legal, regulatory, audit or certification advice; it does not state definitive minimum requirements for any law, standard or regulator; and no control or combination of controls guarantees elimination of risk. Applicability, adequacy and operating effectiveness depend on the specific use case, implementation, jurisdiction, organisation, threat model and evidence. Authoritative requirements and qualified specialists should be used where formal compliance or certification matters.

Core proposition

Getting agents to do what we want is not primarily a prompting problem. It is the problem of translating human intent into a layered system of specification, architecture, authority, validation, testing, oversight, monitoring, intervention and accountability — and then gathering evidence that those layers actually work in the environment where the agent operates.

Notices

Licence. This work is published by Cognostrix under a Creative Commons Attribution 4.0 International licence (CC BY 4.0). You may copy, distribute, adapt and build on it, including commercially, provided Cognostrix is credited. Suggested attribution: *Cognostrix, "Agentic AI Controls, Governance and Assurance — A Reference Catalogue", v1.0, September 2026.*

No claim of originality. This catalogue is a consolidation. The controls it describes are drawn from published standards, regulations, supervisory guidance, vendor documentation and practitioner literature, and are credited to their sources throughout. Cognostrix claims no authorship of the underlying ideas and asserts no rights over them. Where a control has no single identifiable source, that is a statement about the published record, not a claim of invention.

No completeness. The 113 controls are a baseline, not an exhaustive set. Absence of a control here is not evidence that it is unnecessary.

Provided as is. This document is provided without warranty of any kind, express or implied. Cognostrix accepts no liability for any loss or damage arising from its use or from reliance on it. Nothing here creates a professional or advisory relationship.

Third-party names. Standards, frameworks, regulations, products and organisations are named for identification and mapping only. All trade marks are the property of their respective owners. Their inclusion does not imply endorsement of this catalogue by, or any affiliation with, the bodies named.

The control stack at a glance

Stage	Practical question
1. Intent	Define purpose, outcomes, prohibited outcomes, consequence and autonomy.
2. Specification	Turn intent into roles, task contracts, instruction precedence, structured inputs and escalation rules.
3. Architecture	Place hard boundaries and separation in system design, not only in model instructions.
4. Authority & context	Control what the agent may know, remember, access, delegate and change.
5. Execution	Bound loops, retries, budgets, transactions, tool use and code execution while work is happening.
6. Verification	Test components and workflows; validate outputs and actions before relying on them.
7. Observation	Trace decisions, actions, drift, anomalies, feedback and the health of controls in production.
8. Intervention & recovery	Escalate, pause, quarantine, roll back, compensate and continue safely when controls fail.
9. Change	Version, approve and revalidate model, policy, data, tool and configuration changes.
10. Governance & assurance	Assign accountability, assess risk, retain evidence, challenge controls and map applicable obligations.

How this guide differs from existing standards and frameworks

This guide is deliberately complementary to established and emerging AI governance, risk and security material. It doesn't look to replace or compete with them, but rather to provide a practical and useful consolidation and insight into the practical application of them.

- NIST provides the AI Risk Management Framework and GenAI profile and, in 2026, launched an AI Agent Standards Initiative.
- ISO/IEC 42001 provides an AI management-system standard and ISO/IEC 23894 provides AI risk-management guidance.
- The EU AI Act creates legal obligations for defined actors and systems.
- CSA publishes a broad AI Controls Matrix and agentic control-plane material.
- OWASP publishes agentic security risk guidance and an Agent Control Standard focused on runtime control and interoperability, released as an independent open specification (v0.1.0) in May 2026 and incorporated into the OWASP GenAI Security Project in September 2026, currently at v0.1 public preview.
- Vendor guidance from Microsoft, OpenAI and Anthropic adds implementation patterns for agents, guardrails, human intervention and evaluations. [1–17]

The organising proposition here is narrower and more operational: start with the question “How do I get this agent to behave as intended?” then expose the available control levers, when they help, how strongly they constrain behaviour, how to test them, what they do not solve and how they can be selected for a specific use case. If an equivalent widely consolidated adopted catalogue emerges, the appropriate response would be to map to it, reuse it or contribute to it.

Table of Contents

Executive Summary	2
Notices.....	2
The control stack at a glance	3
How this guide differs from existing standards and frameworks	3
Table of Contents	4
Control Catalogue	6
PART I — UNDERSTANDING THE PROBLEM	7
1. About This Guide.....	7
2. Why Do Agents Need Controlling?.....	7
3. What Do We Mean by Agent Governance and Control?	8
PART II — THE CONTROL SYSTEM	9
4. The End-to-End Agent Control Lifecycle.....	9
5. The Main Families of Agent Controls.....	9
Relationship to existing work, by control family.....	10
6. Overarching Principles of Effective Agent Control.....	11
7. Control Strength, Type and Effectiveness.....	12
PART III — USING THE FRAMEWORK	13
8. How to Approach a Control Assessment.....	13
9. Control Assessment Intake & Scoping Questionnaire	13
10. Recommended Assessment Output	15
11. “I Have This Problem — Which Controls Should I Consider?”	15
12. Illustrative Control Baselines by Autonomy and Consequence	16
13. Machine-Assisted Assurance and Governance-Agent Use	16
PART IV — THE CONTROL CATALOGUE	18
14. Control Catalogue Index.....	18
15. Detailed Control Catalogue	21
Purpose & outcome	21
Specification & instruction	23
Architecture & workflow.....	25
Context & data	27
Tools, permissions & authority	29
Behavioural & policy	31
Runtime execution	34
Output validation	35
Testing & evaluation	37
Human oversight	39
Multi-agent systems	42
Observability & monitoring.....	43
Security & adversarial resilience	45

Failure, recovery & intervention 47

Change & configuration 48

Governance & assurance 50

PART V — QUICK-REFERENCE MATRICES.....53

16. Control Family × Primary Objective.....53

17. Control Family × Control Type 53

18. Standards and Regulatory Alignment — Indicative Crosswalk 54

19. EU AI Act — Indicative High-Risk-System Support Map 54

21. Master Control Metadata Table 55

PART VI — APPENDICES 60

Appendix A — Glossary 60

Appendix B — References and Further Reading.....60

Appendix C — Relationship to the Existing Landscape (“Wheel Check”) 61

Appendix D — Dogfooding and Continuous Improvement.....62

Appendix E — Version and Contribution Model 62

Appendix F — First Dogfooding Assessment Template 63

Closing Note 63

Control Catalogue

The detailed control catalogue is intentionally long. Use Part III first to establish assessment context and select candidate controls, then use the Control Catalogue Index, problem-to-control guide, autonomy baselines and standards alignment tables to jump to relevant controls. The compact fallback index below is also provided to assist with navigation.

Section
PART I — UNDERSTANDING THE PROBLEM
1. About This Guide
2. Why Do Agents Need Controlling?
3. What Do We Mean by Agent Governance and Control?
PART II — THE CONTROL SYSTEM
4. The End-to-End Agent Control Lifecycle
5. The Main Families of Agent Controls
6. Overarching Principles of Effective Agent Control
7. Control Strength, Type and Effectiveness
PART III — USING THE FRAMEWORK
8. How to Approach a Control Assessment
9. Control Assessment Intake & Scoping Questionnaire
10. Recommended Assessment Output
11. “I Have This Problem — Which Controls Should I Consider?”
12. Illustrative Control Baselines by Autonomy and Consequence
13. Machine-Assisted Assurance and Governance-Agent Use
PART IV — THE CONTROL CATALOGUE
14. Control Catalogue Index
15. Detailed Control Catalogue
PART V — QUICK-REFERENCE MATRICES
16. Control Family × Primary Objective
17. Control Family × Control Type
18. Standards and Regulatory Alignment — Indicative Crosswalk
19. EU AI Act — Indicative High-Risk-System Support Map
20. EU AI Act — transparency obligations that apply now
21. Master Control Metadata Table
PART VI — APPENDICES
Appendix A — Glossary
Appendix B — References and Further Reading
Appendix C — Relationship to the Existing Landscape (“Wheel Check”)
Appendix D — Dogfooding and Continuous Improvement
Appendix E — Version and Contribution Model
Appendix F — First Dogfooding Assessment Template
Closing Note

PART I — UNDERSTANDING THE PROBLEM

1. About This Guide

This is a practical reference for people designing, building, testing, deploying, operating, governing or auditing AI agents and agentic workflows. It aims to make the available levers visible without implying that every lever belongs in every system.

Ways to use the guide

Mode	How to use it
Read it	Use Parts I and II to understand the control problem, the lifecycle and the overarching principles.
Look something up	Use the catalogue index and stable control IDs to find one mechanism and understand its implementation, evidence and limitations.
Solve a problem	Start with the assessment intake or quick-reference matrices, identify relevant risks and candidate controls, then inspect the detailed control cards before deciding what is proportionate.
Audit an existing AI deployment	Evaluate an existing AI deployment to determine the risks and potential mitigations.
Plan a new AI deployment	Use the guide to determine which controls should be applied, and the related design, build, test artefacts and actions required.

Note that all of the above activities could be conducted manually and/or in an automated manner using AI itself. PART 111 of this document goes into more detail on both the manual and automated approaches.

Who it is for

Agent developers, software and enterprise architects, product owners, engineering leaders, testers, red teams, security teams, model-risk and operational-risk functions, compliance and legal teams, internal audit, AI governance functions, executives accountable for deployment, and governance or assurance agents operating under human accountability.

What it is not

- A substitute for engineering judgement or domain expertise.
- A new certification standard or a complete compliance framework.
- A requirement to implement all 113 controls.
- A guarantee that probabilistic systems can be made deterministic.
- Proof that a risk is controlled merely because a control is documented.
- A static document that should remain unchanged as practice and threats evolve.

Before selecting controls: establish the assessment context

Control selection without context tends to become checklist compliance. Before recommending controls, establish what the system is for, what it can affect, how autonomous it is, what it can access, which obligations apply, how failure can be detected and reversed, and what evidence is available. Section 9 provides a reusable 50-question Control Assessment Intake & Scoping Questionnaire for humans and governance agents.

2. Why Do Agents Need Controlling?

The core problem is not that agents are inherently uncontrollable. It is that the mapping from human intention to agent behaviour passes through several uncertain layers: language, model interpretation, context, tools, data, architecture, permissions, external systems and other agents. Traditional software also fails, but agentic systems add flexible interpretation and action selection to the failure surface.

Why expectation can diverge	Control implication
Natural language is under-specified	People omit assumptions, organisational knowledge, exceptions and priorities because other people infer them.
Behaviour is probabilistic	The same broad task can be solved through different reasoning and action paths, some of which were not anticipated.
Stated intent differs from intended outcome	An agent can optimise what was written rather than what the organisation actually wanted.

Why expectation can diverge	Control implication
Context is heterogeneous and sometimes hostile	Retrieved documents, user content, web pages and tool results can be stale, contradictory, malicious or simply wrong.
Errors compound over long horizons	Small planning or interpretation errors can propagate through multi-step workflows and become material only later.
The environment changes	Data, policies, APIs, vendors, models and business rules change after testing.
Tools convert words into consequences	A mistaken sentence is one thing; the same mistake attached to payments, code deployment, email or records can change external state.
Delegation creates new paths	Sub-agents and tools can create authority chains and interactions the original designer did not explicitly reason about.
Models and vendors change	A stable API or model name does not necessarily mean stable behaviour, capability, security or policy.
Humans are part of the system	Reviewers can rubber-stamp, misunderstand, become overloaded or over-trust fluent explanations.

Recurring principle

When a boundary is genuinely non-negotiable, enforce it in a layer capable of enforcing it. A prompt can influence behaviour; it is not equivalent to a permission boundary, transaction limit, schema validator, policy engine or network control.

3. What Do We Mean by Agent Governance and Control?

Term	Working definition in this guide
Governance	The authority, accountability, decision rights and rules that determine what the organisation permits, requires and owns.
Control	A mechanism that prevents, constrains, detects, verifies, corrects or recovers from undesirable behaviour or conditions.
Assurance	Evidence and reasoned confidence that the system and its controls are designed appropriately, implemented and operating as intended.
Oversight	Observation, review, challenge, approval and intervention around the system's operation.
Risk management	The process of identifying uncertainty and potential harm, assessing significance, selecting treatment and deciding residual risk.
Compliance / conformance	Meeting applicable legal, regulatory, contractual, policy or standards requirements. The authoritative requirement defines what must be met; this guide can only support implementation and evidence.

Useful shorthand

Governance decides what should happen. Controls help make it happen. Assurance provides evidence that it is happening. Oversight creates observation and intervention. Risk management determines where attention and control strength should be concentrated.

PART II — THE CONTROL SYSTEM

4. The End-to-End Agent Control Lifecycle

The lifecycle is a navigation model, not a claim that governance happens only at one stage. Governance, security, risk and assurance overlay the entire system. A control may also act at several lifecycle stages.

Lifecycle stage	Core question	Representative controls
1. Intent	Why does the agent exist? What outcome, prohibited outcome, consequence and autonomy are acceptable?	PUR-01 to PUR-06; GOV-02
2. Specification	What exactly is expected? Which instructions, inputs, assumptions and escalation rules govern the task?	SPEC-01 to SPEC-07; MULTI-01
3. Architecture	Where should flexibility be permitted and where should deterministic process, separation or reversibility constrain it?	ARCH-01 to ARCH-07; MULTI-04
4. Authority & context	What may the agent know, retrieve, remember, access, delegate and change?	CTX-01 to CTX-09; AUTH-01 to AUTH-08; MULTI-02; MULTI-03
5. Execution	How are behaviour, policy, runtime resources, tool calls and transactions constrained during operation?	BEH-01 to BEH-08; RUN-01 to RUN-07; MULTI-06
6. Verification	How do we test and validate outputs, actions and controls before relying on them?	OUT-01 to OUT-06; TEST-01 to TEST-10; MULTI-05
7. Observation	Can we see what happened, detect anomalies and drift, and know whether controls themselves are healthy?	OBS-01 to OBS-07
8. Intervention & recovery	How can people and systems escalate, stop, contain, reverse and continue when something goes wrong?	HUM-01 to HUM-07; FAIL-01 to FAIL-06
9. Change	How are models, prompts, tools, policies, dependencies and configurations changed without invalidating prior assurance?	CHG-01 to CHG-05
10. Governance & assurance	Who owns the risk, what evidence supports confidence, and how are external and internal obligations mapped?	GOV-01 to GOV-08

5. The Main Families of Agent Controls

Control family	What it primarily governs	Baseline coverage
Purpose & outcome	Defines what the agent is for, what success means, what must never happen, and how much autonomy is justified.	6 controls: PUR-01–PUR-06
Specification & instruction	Reduces ambiguity through role definitions, task contracts, instruction hierarchy, structured inputs and traceable policies.	7 controls: SPEC-01–SPEC-07
Architecture & workflow	Places critical controls in system design: deterministic boundaries, explicit state, separation of duties and reversible execution.	7 controls: ARCH-01–ARCH-07
Context & data	Controls what the agent may know, retrieve, remember, infer and expose, and the provenance and quality of that context.	9 controls: CTX-01–CTX-09
Tools, permissions & authority	Constrains what the agent can access or change, under whose identity, for how long and within what transactional limits.	8 controls: AUTH-01–AUTH-08
Behavioural & policy	Shapes model behaviour where deterministic enforcement is not practical, including policy reasoning, abstention and disclosure.	8 controls: BEH-01–BEH-08

Control family	What it primarily governs	Baseline coverage
Runtime execution	Bounds execution time, cost, retries, loops, actions and concurrency while the agent is operating.	7 controls: RUN-01–RUN-07
Output validation	Checks structure, policy, facts, code and downstream compatibility before outputs are trusted or acted upon.	6 controls: OUT-01–OUT-06
Testing & evaluation	Builds evidence through component, regression, scenario, adversarial, integration, UAT, shadow and release-gate evaluation.	10 controls: TEST-01–TEST-10
Human oversight	Introduces meaningful human review, approval, escalation, intervention and production sampling where judgement is needed.	7 controls: HUM-01–HUM-07
Multi-agent systems	Controls communication, identity, delegation, orchestration, independent challenge and conflict among interacting agents.	6 controls: MULTI-01–MULTI-06
Observability & monitoring	Makes behaviour, decisions, drift, incidents and the health of controls visible in production.	7 controls: OBS-01–OBS-07
Security & adversarial resilience	Applies agent-specific and conventional cyber controls against injection, credential theft, supply-chain compromise and abuse.	6 controls: SEC-01–SEC-06
Failure, recovery & intervention	Provides stop, safe-state, rollback, incident-response and continuity mechanisms when prevention fails.	6 controls: FAIL-01–FAIL-06
Change & configuration	Ensures behavioural changes are versioned, approved, revalidated and assessed for external dependency impact.	5 controls: CHG-01–CHG-05
Governance & assurance	Creates ownership, risk decisions, evidence, independent challenge and mapping to applicable obligations.	8 controls: GOV-01–GOV-08

Relationship to existing work, by control family

The mechanisms in this catalogue are drawn from established practice. External review identified, for each family, the bodies of work it most closely relates to; those are summarised below. This is a map of adjacency, not an assessment: the sources named have not been evaluated here, and the summaries describe fields rather than specific documents.

Much of what follows is conventional. Four families — runtime execution, output validation, failure and recovery, and change and configuration — are largely standard engineering practice applied to a new kind of system, and the catalogue says so in the cards themselves rather than claiming otherwise.

Family	Related published work
Purpose & outcome	AI management-system policy and impact-assessment controls; governance actions in the NIST generative-AI profile; the CSA controls matrix's governance domain. Risk-tiering and autonomy levels echo published autonomy-level frameworks for agentic AI and the academic literature on autonomy grading.
Specification & instruction	AI management-system lifecycle controls; published instruction-hierarchy work; OWASP's prompt-injection and goal-hijack categories. Parameter validation is conventional input validation, plus OWASP's tool-misuse category.
Architecture & workflow	Published secure-AI design-pattern frameworks; the "agents rule of two" pattern; the LLM-as-judge literature underlying generation–verification separation; conventional fail-safe engineering.
Context & data	OWASP's memory-poisoning category; the academic memory-attack literature; MITRE ATLAS's agentic memory techniques; conventional data-provenance practice.
Tools, permissions & authority	OWASP's identity-and-privilege-abuse category; conventional access control; the emerging agent-identity standards from major identity vendors and the IETF. Segregation of duties is classic internal-control practice.
Behavioural & policy	Constitutional-AI work; the guardrail literature; the deterministic policy-engine pattern as it appears in vendor agent-governance tooling.
Runtime execution	OWASP's resource-exhaustion and runaway categories; conventional container isolation; standard distributed-systems engineering.
Output validation	OWASP's insecure-output-handling category; conventional static, dynamic and composition analysis.

Family	Related published work
Testing & evaluation	Measurement actions in the NIST generative-AI profile; OWASP and MITRE red-team guidance; conventional software testing for the boundary and regression controls.
Human oversight	Statutory human-oversight obligations; AI management-system human-oversight controls; the human-factors literature on automation bias — which is comparatively under-represented in security-oriented control catalogues.
Multi-agent systems	OWASP's inter-agent-communication and rogue-agent categories; the multi-agent security literature; the same agent-identity standards as above.
Observability & monitoring	OWASP's agent control standard, including its observability work and agent bill of materials; conventional audit and accountability controls; drift monitoring from MLOps practice. Control-health monitoring is comparatively under-published.
Security & adversarial resilience	OWASP's LLM and agentic top-tens; MITRE ATLAS. Several controls here are conventional identification, communications-protection and risk-assessment practice, and one says so in its own text.
Failure, recovery & intervention	Conventional incident-response and contingency planning. The agent-specific incident playbook carries some novelty; the remainder does not.
Change & configuration	Conventional configuration management. One control carries agent-specific content; the rest is standard practice.
Governance & assurance	AI management-system clauses; the govern function of the NIST AI risk framework; current internal-audit standards; the model-risk regimes. Closure-verification review is the exception in this family.

6. Overarching Principles of Effective Agent Control

The catalogue contains individual mechanisms. The following principles help combine those mechanisms into an effective system rather than a collection of isolated controls.

Defence in depth. Do not depend on one prompt, one guardrail, one reviewer or one policy. Combine controls that fail differently.

Hard boundaries need hard enforcement. If an outcome absolutely must not occur, enforce the boundary in permissions, policy, architecture or deterministic code wherever possible.

Authority should be proportional to consequence. The more consequential the action, the stronger the permission, verification, approval and recovery mechanisms should be.

Least privilege by default. Give the agent, tool and delegated identity only the access required for the current task, for only as long as required.

Separate generation from verification. Do not make a single probabilistic generation path the only judge of its own correctness.

Make uncertainty actionable. Define when uncertainty causes clarification, abstention, alternative evidence, human escalation or reduced authority.

Fail safely and prefer reversible actions. Design for deny, queue, draft, stage, pause, rollback or compensate when confidence or control availability is insufficient.

Make important behaviour observable. A control that cannot be evidenced, monitored or reconstructed is difficult to assure.

Test controls, not only agents. Verify that the mechanism itself blocks, detects or recovers from the failure it claims to address.

Assume correlated failure. Two agents, two prompts or two model-based judges may share the same blind spot. Seek genuinely independent controls for high-consequence risks.

Human oversight must be meaningful. A human approval step is weak if the reviewer lacks time, information, expertise or authority to disagree.

More autonomy requires more assurance. As planning freedom, permissions, persistence, delegation and consequence increase, so should evidence, monitoring and intervention capability.

Conventional security still applies. Agent-specific controls add to, not replace, identity, network, vulnerability, secrets, logging and incident-response disciplines.

Changes can invalidate prior assurance. Model, prompt, policy, tool, data and vendor changes can alter behaviour and should trigger proportionate impact assessment and revalidation.

Treat this framework as living. New agent architectures, failures, attacks and regulatory interpretations will create new control mechanisms and change the relative value of existing ones.

7. Control Strength, Type and Effectiveness

7.1 Typical control strength

Typical strength	Meaning
Advisory	Provides guidance or a signal but does not materially constrain execution by itself. Advisory describes a rating a control degrades to at runtime — for example a permission control operating under a person’s own credentials — rather than a tier a control is authored at. No card in this catalogue is rated Advisory as designed.
Behavioural	Uses model instructions, prompting, examples or reasoning patterns to influence behaviour probabilistically.
Procedural	Relies materially on people, process, ownership, review or organisational discipline.
Enforced	Uses a technical or workflow mechanism that can block, constrain, route or require an action independently of the model’s preference.
Deterministic	Imposes a rule or system boundary with predictable machine enforcement for the defined condition, such as a permission check, schema validator or hard transaction ceiling.

“Strength” is contextual, not a quality score. A deterministic control can be badly configured; a procedural control can be highly effective; and a behavioural control can be appropriate for low-consequence judgement. A strength value describes the mechanism a control relies on, not a guarantee about its outcome: a control rated Enforced or Deterministic can still fail, and where a card has a known boundary it is stated on the card itself. Deterministic strength assumes the agent acts under an identity distinct from the humans it works for. Where an agent executes under a person’s own credentials, permission- and identity-based controls constrain other people, not the agent, and should be rated advisory regardless of how they are configured. The field describes the mechanism’s typical mode of constraint, not guaranteed effectiveness.

7.2 Control type

Type	Purpose
Preventive	Reduce the probability that an undesirable condition or action occurs.
Detective	Identify failure, misuse, drift, non-conformance or control degradation.
Corrective	Act after detection to contain, remediate or change the situation.
Recovery	Restore an acceptable state or service after a failure or intervention.

7.3 Do not confuse “a control exists” with “the risk is controlled”

Every material assessment should separate the following dimensions. This is equally important for a human audit and for a future governance agent:

Assessment dimension	Question to answer	Illustrative rating
Design adequacy	If implemented as designed, is this control suitable and proportionate for the risk and use case?	Adequate / Partly adequate / Inadequate / Not applicable
Implementation status	Has the design actually been implemented in the relevant environment and scope?	Implemented / Partial / Planned / Absent
Operating effectiveness	Does evidence show that it works consistently in practice, including relevant failure modes?	Effective / Partly effective / Ineffective / Not tested
Evidence confidence	How strong, current, representative and independent is the evidence supporting the claim?	High / Medium / Low / None
Residual risk	What risk remains after considering control design and demonstrated effectiveness?	Accept / Remediate / Escalate / Avoid / Transfer, using the organisation’s own risk method

Assessment discipline

A documented policy, prompt or approval step is not evidence by itself that a risk has been controlled. The assessment should distinguish design, implementation and demonstrated operation, and state uncertainty where evidence is weak.

PART III — USING THE FRAMEWORK

8. How to Approach a Control Assessment

Control selection should be a context-driven risk assessment, not a checklist exercise. The catalogue is a library of candidate mechanisms; the assessment determines which mechanisms are relevant, proportionate and evidenced for the use case. A practical sequence is:

1. Establish context — purpose, users, consequence, reversibility, autonomy, data, authority, dependencies and applicable obligations.
2. Identify scenarios — normal failure, misuse, adversarial behaviour, dependency failure and control failure; assess inherent consequence using the organisation's risk method.
3. Select candidate controls — use the problem map, autonomy/consequence baselines, standards crosswalk and control metadata to identify mechanisms worth investigating.
4. Assess the controls that actually exist — separate design adequacy, implementation status, operating effectiveness and evidence confidence rather than treating documentation as proof.
5. Decide and iterate — state residual risk and uncertainty, prioritise remediation, obtain accountable acceptance where required, and reassess when the system or its environment changes.

9. Control Assessment Intake & Scoping Questionnaire

This questionnaire is the recommended starting point before selecting or evaluating controls. Its purpose is to make the context explicit enough for a reviewer — human or machine-assisted — to reason about consequence, autonomy, attack surface, obligations, evidence and recovery. It is intentionally broader than technical architecture.

How a governance agent should use this intake

Ask only the questions whose answers are not already evidenced. Record the answer, source, confidence, assumptions and unresolved items. Do not silently infer high-impact facts such as regulatory applicability, consequence, legal effect or authorised risk acceptance. When material information is unavailable, request it where practicable; otherwise report the uncertainty and adjust the assessment accordingly.

A. Purpose, users and outcomes

- What is the agent or agentic workflow intended to achieve, and what is explicitly outside scope?
- Who are the users, subjects, customers or other people affected by its outputs or actions?
- What does a successful outcome look like, and which measures or acceptance criteria demonstrate it?
- Which outcomes or actions are prohibited even if they appear to help achieve the stated objective?
- Is the agent advisory, decision-supporting, decision-making, or directly executing actions?

B. Consequence, reversibility and scale

- What is the plausible consequence of an incorrect, delayed, manipulated or unauthorised output or action?
- Could the agent create legal, financial, safety, employment, credit, privacy, security or other material effects for people or organisations?
- How many users, transactions, records, systems or downstream decisions could a single failure affect?
- Can an incorrect action be reversed, compensated for or contained after it occurs?
- What is the maximum tolerable loss, exposure, delay or harm before mandatory human intervention is required?

C. Autonomy, actions and delegation

- How independently may the agent plan, choose methods, select tools and decide when a task is complete?
- Can it create, modify, delete, send, approve, purchase, trade, publish, deploy or otherwise change external state?
- Can it delegate tasks to other agents or services, and can those downstream components take consequential actions?
- What authority must remain with a human or deterministic system regardless of agent confidence?
- What limits should apply to action value, rate, volume, duration, scope, delegation depth or concurrency?

D. Data, context, memory and knowledge

- What data does the agent receive, retrieve, infer, generate and store, and how sensitive or regulated is it?

- Which sources are authoritative, trusted, untrusted, user-controlled or externally retrieved?
- Does the agent require memory across turns or sessions; what may be remembered, for how long and for which identity?
- Can data from one user, tenant, case or security domain become visible to another through context, memory, retrieval or logs?
- How will the system handle stale, contradictory, incomplete, manipulated or unavailable information?

E. Tools, permissions, identity and external systems

- Which tools, APIs, databases, file systems, websites, code runtimes or physical systems can the agent access?
- Which permissions are read-only versus write or execute, and can they be narrowed to specific resources or actions?
- Under whose identity does each action occur: the user, the agent, a service account or another delegated identity?
- How are credentials granted, scoped, rotated, revoked and prevented from entering prompts, memory or logs?
- Which external systems or actions are high consequence enough to require deterministic policy or approval before execution?

F. Architecture, dependencies and multi-agent behaviour

- Which steps genuinely require model judgement, and which can be implemented more safely as deterministic code, rules or workflow?
- What are the allowed process states and transitions, and can the agent bypass a required control or approval step?
- What models, vendors, tools, connectors, protocols, libraries and data providers does the system depend on?
- If multiple agents interact, how are identity, message semantics, authority, delegation, conflicts and deadlocks controlled?
- What common dependencies could cause several supposedly independent controls or agents to fail together?

G. Human oversight, accountability and user experience

- Who is accountable for the agent, its business outcome, technical operation, risk decisions and incidents?
- Which decisions require human review before action, and which can be reviewed after the fact through sampling?
- Does the reviewer have the information, expertise, time and authority needed to challenge the agent meaningfully?
- When must the agent disclose uncertainty, AI involvement, limitations, evidence or the need for human judgement?
- How can users correct, appeal, report or stop an agent decision or action when something appears wrong?

H. Standards, law, regulation, policy and contracts

- Which jurisdictions, laws, regulations or sector rules may apply to this use case and the people it affects?
- Are there specific standards or frameworks the organisation has committed to, such as ISO/IEC 42001, NIST AI RMF, CSA AICM or internal control frameworks?
- Do customer, supplier, employment, confidentiality, data-processing, outsourcing or service contracts impose relevant obligations?
- Are there internal policies, risk appetites, model-risk rules, security standards or audit expectations that constrain the design?
- Who is authorised to interpret applicability and sign off residual legal, regulatory or contractual risk?

I. Testing, evidence and release

- What failure and misuse scenarios must be tested before release, including rare but severe outcomes?
- What component, integration, adversarial, end-to-end, UAT, shadow or parallel-running evidence is proportionate to the risk?
- What are the release thresholds for quality, safety, security, reliability, latency and cost?
- Can every material control claim be linked to evidence showing design, implementation and operating effectiveness?
- Which changes require re-testing or re-approval before they may reach production?

J. Production monitoring, change and recovery

- Which production signals will reveal quality failure, unusual behaviour, drift, control degradation, abuse or escalating risk?
- Can consequential actions be reconstructed after the fact, including effective instructions, model, tools, identity, approvals and evidence?
- How quickly can the organisation pause, quarantine or disable the agent or a specific capability?
- What fallback, rollback, compensating action, incident response and business-continuity mechanisms are required?
- How will incidents, near misses, user feedback and model or vendor changes feed back into tests, controls and the risk assessment?

10. Recommended Assessment Output

The intake should result in a traceable risk-and-control assessment rather than a generic list of “best practices.” A useful assessment record contains at least the following fields:

Assessment field	What to capture
Use case / requirement	The specific capability, workflow, requirement or change being assessed.
Failure or misuse scenario	What could go wrong, including accidental, adversarial and control-failure scenarios.
Consequence and affected parties	Who or what could be affected; scale, reversibility and potential impact.
Inherent risk	The organisation’s assessment before considering existing controls.
Applicable obligations	Relevant legal, regulatory, contractual, standards and internal-policy considerations, with source/owner.
Existing controls	Stable control IDs plus implementation-specific mechanisms already present.
Design adequacy	Whether those controls are appropriate and proportionate for the scenario.
Implementation status	Whether the intended design exists in the assessed environment.
Operating effectiveness	Whether current evidence demonstrates the control works in practice.
Evidence	Tests, configuration, logs, approvals, incident history, sampling or other artefacts supporting the conclusion.
Residual risk	Risk remaining after demonstrated controls, using the organisation’s own risk method.
Gap / remediation	Missing, weak or unnecessary controls; action, priority, owner and target state.
Assumptions and uncertainty	Facts inferred, unknown or disputed that could materially change the conclusion.
Decision	Accept, remediate, escalate, avoid, transfer, defer or other organisation-defined disposition, with accountable approver.

A compact risk/control matrix

Scenario	Inherent risk	Existing control IDs	Design	Implemented	Operating effectiveness	Evidence confidence	Residual risk	Action
Example: agent sends an externally visible message to the wrong recipient	High	AUTH-01; AUTH-06; HUM-02; OBS-02	Adequate	Partial	Not yet tested	Low	High	Complete recipient allowlist and approval test before release

The example above is illustrative only. Organisations should use their own risk scales, acceptance authorities and evidence standards.

11. “I Have This Problem — Which Controls Should I Consider?”

This table is a navigation aid, not a prescribed minimum set. Start with the scenario, inspect the referenced controls, then select or reject them based on the intake, consequence, architecture and evidence.

Problem / concern	Controls to investigate first
The agent works outside its intended scope	PUR-01; SPEC-01; SPEC-03; BEH-02; TEST-03; OBS-04
Its answers sound plausible but factual grounding is weak	CTX-02; CTX-03; CTX-06; BEH-05; OUT-03; HUM-01
The agent has too much authority	PUR-05; AUTH-01; AUTH-02; AUTH-04; AUTH-06; AUTH-08; HUM-02
A document, web page or tool result may contain malicious instructions	SPEC-02; CTX-01; CTX-02; SEC-01; AUTH-01; TEST-05
The agent can take irreversible or high-value actions	ARCH-07; AUTH-06; AUTH-08; RUN-05; HUM-02; FAIL-03
We cannot explain what happened after an incident	SPEC-07; OBS-01; OBS-02; CHG-01; GOV-03
The agent loops, retries or consumes too much cost	ARCH-04; RUN-01; RUN-02; RUN-03; RUN-04; OBS-03
The agent can generate or execute code	RUN-07; OUT-04; TEST-01; TEST-05; SEC-05; HUM-01
Multiple agents delegate work to each other	MULTI-01; MULTI-02; MULTI-03; MULTI-04; MULTI-06; OBS-01
Human reviewers are likely to rubber-stamp	HUM-04; HUM-06; OUT-03; TEST-09; OBS-06
A model, prompt or tool update may silently break behaviour	SPEC-07; TEST-02; TEST-07; CHG-01; CHG-03; CHG-04; OBS-05
We need to stop and recover from bad behaviour quickly	HUM-05; FAIL-01; FAIL-02; FAIL-03; FAIL-04; OBS-04

Problem / concern	Controls to investigate first
The agent handles sensitive or confidential data	CTX-01; CTX-07; AUTH-05; SEC-02; SEC-04; OBS-02
A third-party model, connector or dependency could change or be compromised	SEC-03; CHG-04; AUTH-02; AUTH-03; OBS-04; GOV-04
The system will face audit or regulatory scrutiny	PUR-04; GOV-01; GOV-02; GOV-03; GOV-04; OBS-02; TEST-08

12. Illustrative Control Baselines by Autonomy and Consequence

Important caveat

The profiles below are starting points for discussion only. They are not regulatory, certification or safety minima. A low-autonomy system can still be high risk because of sensitive data or scale, while a highly autonomous sandbox may be relatively low consequence. Always use the assessment context.

Illustrative profile	Typical characteristics	Controls/families that become increasingly important
Profile A — Assistive / low consequence	Produces suggestions or drafts; no independent write authority; outputs are easily reviewed and reversible.	Purpose and role specification; data boundaries; output validation proportionate to task; basic evals; logging; user feedback; ownership and change control.
Profile B — Bounded workflow / moderate consequence	Uses tools inside a defined workflow and may modify low-impact internal state.	Profile A plus explicit workflow, least privilege, tool allowlists, structured actions, runtime limits, integration tests, meaningful human escalation, anomaly and control-health monitoring, rollback/fallback.
Profile C — Consequential autonomous action	Can execute externally visible, financial, legal, security-sensitive or otherwise material actions within defined limits.	Profile B plus transaction/value ceilings, stronger identity and delegated authority, independent policy/action validation, approval gates where warranted, adversarial testing, shadow/parallel evidence, incident playbooks, robust audit trails and periodic assurance.
Profile D — Persistent or multi-agent high autonomy	Plans over long horizons, delegates, persists state, coordinates agents or can change significant external state.	Profile C plus explicit delegation attenuation, authenticated inter-agent contracts, supervisor controls, conflict/deadlock rules, tighter budgets/timeouts, stronger egress/supply-chain controls, continuous control-health/drift monitoring, kill/quarantine mechanisms and frequent independent challenge.

13. Machine-Assisted Assurance and Governance-Agent Use

The catalogue is deliberately structured so it can later serve as a reference corpus for a persistent governance or assurance agent. Such an agent could interview the system owner using the intake, inspect requirements and evidence, map risks to candidate controls, identify gaps and produce a first-pass risk/control matrix for human review. The agent should support accountable decision-makers, not become an unreviewed authority for legal applicability, risk acceptance or certification.

Illustrative governance-agent workflow

1. Ingest the requirement, architecture, agent brief, control descriptions and available evidence.
2. Run the assessment intake, asking only material unanswered questions and recording uncertainty.
3. Identify plausible failure, misuse, compromise and control-failure scenarios.
4. Use control metadata to retrieve candidate mechanisms by risk, lifecycle, autonomy, strength and implementation context.
5. Inspect implementation evidence and rate design adequacy, implementation status, operating effectiveness and evidence confidence separately.
6. Map applicable standards, regulatory or policy obligations only where authoritative sources and applicability are available; flag specialist decisions rather than guessing.
7. Estimate residual risk using the organisation's approved method and propose prioritised remediation with stable control IDs.
8. Present assumptions, disagreements and decisions requiring human ownership; retain the assessment as an auditable artefact.
9. Feed incidents, near misses, releases and review findings back into the catalogue, tests and system assessment.

Control metadata schema for later export

Field	Purpose
control_id	Stable identifier, e.g. AUTH-06
Name	Canonical control name
Family	Primary catalogue family
Definition	What the control is and what it seeks to influence
risks_addressed	Searchable risk/failure tags
delivery_stages	Where the control acts
control_type	Preventive, detective, corrective and/or recovery
typical_strength	Advisory, behavioural, procedural, enforced or deterministic
autonomy_relevance	Indicative relevance as autonomy/consequence increases
implementation_effort	Indicative implementation effort
implementation_patterns	Practical design/operating approaches
test_methods	Ways to challenge the control
evidence_types	Evidence that can support implementation/effectiveness claims
Limitations	Known gaps and what the control does not solve
Complements	Controls that commonly strengthen the mechanism
standards_alignment	Indicative mappings to external obligations/frameworks
catalogue_version	Version in which the definition/mapping applies

A later machine-readable JSON/YAML or control-database representation should preserve these fields and add mapping confidence, source/version/date, organisation-specific implementation, assessment status and evidence links. The Word document can remain the human-readable reference version.

PART IV — THE CONTROL CATALOGUE

14. Control Catalogue Index

Every control has one canonical ID and home. The same control may appear in many matrices because controls are multidimensional. Stable IDs allow people, test artefacts, architecture records and governance agents to refer to the same mechanism without duplicating definitions.

ID	Control	Family	Typical strength	Autonomy relevance
PUR-01	Intended purpose and scope boundary	Purpose & outcome	Procedural	All
PUR-02	Success criteria and measurable outcomes	Purpose & outcome	Procedural	All
PUR-03	Prohibited outcomes and non-negotiable boundaries	Purpose & outcome	Enforced	All
PUR-04	Risk and impact tiering	Purpose & outcome	Procedural	All
PUR-05	Explicit autonomy level	Purpose & outcome	Procedural	All
PUR-06	Differential outcomes across groups	Purpose & outcome	Procedural	All
SPEC-01	Agent brief / role specification	Specification & instruction	Behavioural	All
SPEC-02	Instruction hierarchy and precedence	Specification & instruction	Behavioural	All
SPEC-03	Task contract and acceptance criteria	Specification & instruction	Behavioural	All
SPEC-04	Examples and counterexamples	Specification & instruction	Behavioural	All
SPEC-05	Structured input contracts	Specification & instruction	Deterministic	All
SPEC-06	Ambiguity and uncertainty escalation rules	Specification & instruction	Behavioural	All
SPEC-07	Instruction and policy versioning / traceability	Specification & instruction	Enforced	All
ARCH-01	Deterministic-versus-agentic boundary	Architecture & workflow	Deterministic	All
ARCH-02	Explicit workflow / state machine	Architecture & workflow	Deterministic	Medium+
ARCH-03	Planner-executor separation	Architecture & workflow	Enforced	Medium+
ARCH-04	Bounded task decomposition and delegation	Architecture & workflow	Enforced	Medium+
ARCH-05	Generation-verification separation	Architecture & workflow	Enforced	All
ARCH-06	Safe defaults and fail-closed architecture	Architecture & workflow	Deterministic	Medium+
ARCH-07	Reversible and staged execution design	Architecture & workflow	Enforced	Medium+
CTX-01	Context minimisation	Context & data	Enforced	All
CTX-02	Trusted-source and provenance controls	Context & data	Enforced	All
CTX-03	Grounded retrieval with source citation	Context & data	Enforced	All
CTX-04	Memory scope, retention and expiry	Context & data	Enforced	All
CTX-05	Memory write validation	Context & data	Enforced	Medium+
CTX-06	Freshness and staleness controls	Context & data	Enforced	All
CTX-07	Context, data and tenant isolation	Context & data	Deterministic	All
CTX-08	Personal data in memory and traces	Context & data	Procedural	All
CTX-09	Quality of owned datasets	Context & data	Procedural	All
AUTH-01	Least-privilege authority	Tools, permissions & authority	Deterministic	All
AUTH-02	Tool allowlisting and capability restriction	Tools, permissions & authority	Deterministic	All

ID	Control	Family	Typical strength	Autonomy relevance
AUTH-03	Scoped credentials and agent identity	Tools, permissions & authority	Deterministic	All
AUTH-04	Just-in-time / ephemeral authorisation	Tools, permissions & authority	Deterministic	Medium+
AUTH-05	Resource and destination allowlists	Tools, permissions & authority	Deterministic	All
AUTH-06	Action and transaction limits	Tools, permissions & authority	Deterministic	Medium+
AUTH-07	Tool parameter validation	Tools, permissions & authority	Deterministic	All
AUTH-08	Read/write and maker-checker separation	Tools, permissions & authority	Deterministic	Medium+
BEH-01	Behavioural policy / constitution	Behavioural & policy	Behavioural	All
BEH-02	Scope and relevance guardrail	Behavioural & policy	Enforced	All
BEH-03	Safety and content guardrails	Behavioural & policy	Enforced	All
BEH-04	External deterministic policy engine	Behavioural & policy	Deterministic	Medium+
BEH-05	Abstention / refusal behaviour	Behavioural & policy	Behavioural	All
BEH-06	Uncertainty and confidence handling	Behavioural & policy	Enforced	All
BEH-07	Disclosure of AI interaction	Behavioural & policy	Enforced	All
BEH-08	Reasons for a decision	Behavioural & policy	Procedural	Medium+
RUN-01	Step, loop and action limits	Runtime execution	Deterministic	Medium+
RUN-02	Time, token and cost budgets	Runtime execution	Deterministic	All
RUN-03	Retry limits and backoff	Runtime execution	Deterministic	All
RUN-04	Rate and concurrency limits	Runtime execution	Deterministic	All
RUN-05	Runtime checkpoints / pre-action gates	Runtime execution	Enforced	Medium+
RUN-06	Idempotency and duplicate-action protection	Runtime execution	Deterministic	All
RUN-07	Sandboxed execution environment	Runtime execution	Deterministic	Medium+
OUT-01	Structured output schemas	Output validation	Deterministic	All
OUT-02	Deterministic business-rule validation	Output validation	Deterministic	All
OUT-03	Factual and source verification	Output validation	Enforced	All
OUT-04	Code and executable-artifact validation	Output validation	Enforced	All
OUT-05	Downstream contract and compatibility validation	Output validation	Deterministic	All
OUT-06	Synthetic-content marking	Output validation	Enforced	All
TEST-01	Component and unit tests	Testing & evaluation	Enforced	All
TEST-02	Golden-set regression evaluation	Testing & evaluation	Enforced	All
TEST-03	Scenario and end-to-end system evaluation	Testing & evaluation	Procedural	All
TEST-04	Tool and integration testing	Testing & evaluation	Enforced	All
TEST-05	Adversarial testing and red teaming	Testing & evaluation	Procedural	Medium+
TEST-06	Boundary, negative and misuse-case testing	Testing & evaluation	Procedural	All
TEST-07	Repeated stochastic trials	Testing & evaluation	Procedural	All
TEST-08	Release performance and reliability thresholds	Testing & evaluation	Procedural	All
TEST-09	User acceptance and domain-expert validation	Testing & evaluation	Procedural	All
TEST-10	Shadow, parallel and phased rollout	Testing & evaluation	Procedural	All
HUM-01	Human review of material outputs	Human oversight	Procedural	All

ID	Control	Family	Typical strength	Autonomy relevance
HUM-02	Human approval gate for high-impact actions	Human oversight	Procedural	Medium+
HUM-03	Escalation matrix and handoff	Human oversight	Procedural	All
HUM-04	Meaningful oversight and automation-bias controls	Human oversight	Procedural	Medium+
HUM-05	Human override, cancel and stop capability	Human oversight	Enforced	Medium+
HUM-06	Human quality sampling and periodic review	Human oversight	Procedural	All
HUM-07	Contestability and recourse	Human oversight	Procedural	All
MULTI-01	Explicit inter-agent communication contract	Multi-agent systems	Enforced	Medium+
MULTI-02	Agent identity and authentication	Multi-agent systems	Deterministic	Medium+
MULTI-03	Delegated-authority inheritance and attenuation	Multi-agent systems	Deterministic	High
MULTI-04	Supervisor / orchestrator control	Multi-agent systems	Enforced	High
MULTI-05	Independent critic / checker agent	Multi-agent systems	Behavioural	Medium+
MULTI-06	Conflict, deadlock and consensus rules	Multi-agent systems	Enforced	High
OBS-01	End-to-end tracing and logging	Observability & monitoring	Enforced	All
OBS-02	Decision and action audit trail	Observability & monitoring	Procedural	Medium+
OBS-03	Operational metrics and service-level objectives	Observability & monitoring	Enforced	All
OBS-04	Anomaly and unusual-behaviour detection	Observability & monitoring	Enforced	Medium+
OBS-05	Behaviour, model and data drift monitoring	Observability & monitoring	Procedural	All
OBS-06	User feedback, complaint and near-miss capture	Observability & monitoring	Procedural	All
OBS-07	Control-health monitoring	Observability & monitoring	Enforced	Medium+
SEC-01	Prompt-injection and untrusted-instruction defence	Security & adversarial resilience	Enforced	All
SEC-02	Secrets and sensitive-credential management	Security & adversarial resilience	Deterministic	All
SEC-03	Tool, model and supply-chain trust	Security & adversarial resilience	Procedural	All
SEC-04	Network isolation and egress control	Security & adversarial resilience	Deterministic	Medium+
SEC-05	Vulnerability, patch and hardening management	Security & adversarial resilience	Procedural	All
SEC-06	Abuse, threat and compromise detection	Security & adversarial resilience	Enforced	Medium+
FAIL-01	Pause, kill switch and quarantine	Failure, recovery & intervention	Enforced	Medium+
FAIL-02	Safe-state and fallback behaviour	Failure, recovery & intervention	Enforced	All
FAIL-03	Rollback and compensating actions	Failure, recovery & intervention	Deterministic	Medium+
FAIL-04	Agent-specific incident response playbook	Failure, recovery & intervention	Procedural	All
FAIL-05	Business continuity and manual fallback	Failure, recovery & intervention	Procedural	Medium+
FAIL-06	Regulatory materiality and notification trigger	Failure, recovery & intervention	Procedural	All
CHG-01	Version and configuration management	Change & configuration	Enforced	All
CHG-02	Change approval and segregation	Change & configuration	Procedural	Medium+
CHG-03	Pre-release regression and revalidation	Change & configuration	Enforced	All
CHG-04	Model, vendor, tool and dependency impact assessment	Change & configuration	Procedural	All

ID	Control	Family	Typical strength	Autonomy relevance
CHG-05	Controlled retirement and deprovisioning	Change & configuration	Procedural	All
GOV-01	Accountable ownership, roles and agent inventory	Governance & assurance	Procedural	All
GOV-02	Risk assessment and control-selection rationale	Governance & assurance	Procedural	All
GOV-03	Assurance evidence, periodic review and independent challenge	Governance & assurance	Procedural	Medium+
GOV-04	Standards, legal, regulatory and third-party obligation mapping	Governance & assurance	Procedural	All
GOV-05	Remediation targeting and post-remediation re-rating	Governance & assurance	Procedural	Medium+
GOV-06	Independent validation of agent behaviour	Governance & assurance	Procedural	Medium+
GOV-07	Model-provider dependency and exit	Governance & assurance	Procedural	All
GOV-08	Agent bill of materials	Governance & assurance	Procedural	Medium+

15. Detailed Control Catalogue

How to read a control card

Each entry describes one mechanism. “Typical strength,” “autonomy relevance” and “implementation effort” are orientation metadata, not universal ratings. “How to test / evidence” gives examples rather than an exhaustive assurance programme. “Limitations” is intentionally explicit: no individual control closes the entire risk.

Purpose & outcome

Defines what the agent is for, what success means, what must never happen, and how much autonomy is justified.

PUR-01 — Intended purpose and scope boundary

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Design; Change	Preventive	Procedural	All	Low	misalignment; scope creep; compliance

What it is: Define the business outcome the agent is intended to achieve, the users and environments it serves, and the tasks explicitly outside scope. This is the reference point for judging later behaviour.

How to implement: Write a short intended-purpose statement with in-scope tasks, out-of-scope tasks, affected users, operating context and assumptions. Link it to the agent brief, tests and release criteria.

How to test / evidence: Give reviewers ambiguous and edge-case requests and check whether they can consistently decide whether each is in scope. Test the agent on the same cases.

Limitations / what it does not protect against: A purpose statement is a soft control unless reinforced by permissions, architecture and runtime checks. It can also become stale.

Common complements: SPEC-01; BEH-02; GOV-02

PUR-02 — Success criteria and measurable outcomes

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Test; Operate	Preventive; Detective	Procedural	All	Medium	quality; reliability; misalignment; cost

What it is: Translate intended purpose into observable measures of quality, reliability, latency, cost, safety and user outcome where relevant. Governance is weak when good performance is undefined.

How to implement: Define minimum acceptable thresholds before production. Separate task success from safety constraints and use the same measures in evaluation and production monitoring.

How to test / evidence: Run representative cases and confirm the measures distinguish acceptable from unacceptable behaviour. Check for perverse incentives and hidden tail failures.

Limitations / what it does not protect against: Metrics are proxies and can be gamed or over-optimised. Strong averages can hide severe rare failures.

Common complements: TEST-08; OBS-03; GOV-02

PUR-03 — Prohibited outcomes and non-negotiable boundaries

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Design; Test; Operate	Preventive	Enforced	All	Medium	safety; unauthorised action; compliance; security

What it is: State outcomes, decisions or actions that the agent must never produce or execute. This turns implicit organisational red lines into explicit control objectives.

How to implement: Maintain a concise list and map each prohibition to an enforceable mechanism where possible, such as tool denial, transaction limits, DLP or human approval. Prompt-only prohibitions are not enough for hard boundaries.

How to test / evidence: Pressure the agent to cross each boundary indirectly through delegation, tool calls, transformed inputs and conflicting instructions. Verify the enforcement layer, not only model refusal.

Limitations / what it does not protect against: No list can enumerate every harmful outcome. Overbroad prohibitions can also make the system unusable.

Common complements: AUTH-02; AUTH-06; BEH-04; HUM-02

PUR-04 — Risk and impact tiering

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Govern; Change	Preventive	Procedural	All	Low	accountability; compliance; safety; security

What it is: Classify the use case by the consequence of error, misuse or compromise so autonomy, assurance, oversight and evidence can be proportionate.

How to implement: Use an enterprise scale or low/moderate/high/critical scheme considering financial loss, legal effect, safety, privacy, security, affected people, reversibility and scale. Record the rationale.

How to test / evidence: Have independent reviewers classify sample cases and verify that higher tiers actually trigger stronger controls, testing, approval and monitoring.

Limitations / what it does not protect against: Coarse tiers can hide risk variation. Delivery pressure can bias classification downward.

Common complements: GOV-02; PUR-05; HUM-02; TEST-08

PUR-05 — Explicit autonomy level

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Design; Operate	Preventive	Procedural	All	Low	unauthorised action; misalignment; accountability

What it is: Define how independently the agent may interpret goals, plan, choose tools, act, delegate and persist. Autonomy should be an explicit design decision, not an accidental implementation property.

How to implement: Describe autonomy separately for planning, data access, tool choice, external action, delegation and persistence. Set default ceilings by risk tier. Autonomy ceilings are a risk-appetite decision. Set them where the organisation's risk appetite is owned rather than inside the delivery team, and record who set them: a ceiling chosen by the people building the agent is a statement of capability, not a statement of appetite, and the two are indistinguishable once written down.

How to test / evidence: Exercise tasks that tempt the agent to exceed authorised autonomy, including write actions after read tasks or delegation beyond scope.

Limitations / what it does not protect against: Autonomy is multidimensional and dynamic. A single label does not replace specific permission controls.

Common complements: AUTH-01; AUTH-08; HUM-02; MULTI-03

PUR-06 — Differential outcomes across groups

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Preventive; Detective	Procedural	All	High	discrimination; compliance; accountability; quality

What it is: Establish whether the agent produces materially different outcomes for groups it should not distinguish between, and measure it at the level of the outcome rather than the decision. Human-oversight controls address whether a reviewer can challenge the agent; they do not address whether the agent is worse for some people than others.

How to implement: State, before deployment, which groups the outcome must not differ across and on what basis that list was chosen. Measure at the outcome, not at each step: an agent's path — which tools it called, what it retrieved, whether it escalated or abstained — can be individually defensible at every point and still compose into a differential result. Where the agent abstains or escalates at different rates across groups, treat that as an outcome, not as a safety behaviour. Where a conduct regime applies to the firm, this measurement is the evidence that regime will ask for. Conduct obligations are stated as outcomes for customers, and an agent reaches a differential outcome by composing steps that are each individually defensible — so the exposure arrives by a route on which no single step would raise a flag. Point at the organisation's own conduct framework rather than restating it here.

How to test / evidence: Compare outcomes across the stated groups on production or shadow traffic, including abstention and escalation rates. Evidence is the comparison, the groups it covered, and — stated explicitly — the groups it could not cover because the data to do so does not exist. The absent groups are the finding as much as the measured ones.

Limitations / what it does not protect against: Measuring disparity does not explain or remove it. Some group data cannot lawfully or ethically be collected, which bounds what can be measured and should be recorded rather than left implicit. Parity on measured groups says nothing about unmeasured ones.

Common complements: PUR-02; HUM-04; TEST-03; CTX-09

Specification & instruction

Reduces ambiguity through role definitions, task contracts, instruction hierarchy, structured inputs and traceable policies.

SPEC-01 — Agent brief / role specification

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Change	Preventive	Behavioural	All	Low	misalignment; scope creep; accountability

What it is: Create a persistent, versioned specification of the agent's role, objectives, responsibilities, boundaries, inputs, outputs, decision rights, escalation routes and assumptions.

How to implement: Keep the brief concise but complete. Separate mandatory constraints from helpful context, reference authoritative policies and make the brief available to builders, testers, operators and runtime.

How to test / evidence: Create role-boundary scenarios. Review incidents against the brief: the agent violated it, the brief was unclear, or the brief omitted a requirement.

Limitations / what it does not protect against: A brief is interpreted probabilistically and cannot enforce hard limits by itself. Large briefs can dilute attention or contain contradictions.

Common complements: PUR-01; SPEC-02; TEST-03; GOV-01

SPEC-02 — Instruction hierarchy and precedence

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Operate	Preventive	Behavioural	All	Medium	prompt injection; misalignment; security

What it is: Define which instructions win when system rules, developer rules, policy, user requests, retrieved content and tool outputs conflict.

How to implement: Separate trusted instructions from untrusted content. Tell the agent how conflicts are handled, but enforce high-risk precedence outside the model where possible.

How to test / evidence: Inject conflicting instructions at multiple layers, including documents, web pages and tool results, and verify trusted policy wins.

Limitations / what it does not protect against: Models can still follow salient lower-trust content. Prompt hierarchy alone is not a security boundary.

Common complements: SEC-01; BEH-04; CTX-02

SPEC-03 — Task contract and acceptance criteria

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test	Preventive	Behavioural	All	Low	quality; misalignment; hallucination

What it is: For material tasks, define outcome, required inputs, constraints, deliverable format, completion conditions and acceptance criteria. This narrows the interpretation space.

How to implement: Represent task contracts structurally where possible. Require prerequisite checks and explicit assumptions. Persist long-running task contracts in state rather than conversational memory.

How to test / evidence: Run cases with missing inputs, ambiguous criteria and conflicting constraints. Confirm the agent does not silently invent requirements.

Limitations / what it does not protect against: Poorly specified contracts can create false confidence. Open-ended work cannot always be reduced to objective acceptance criteria.

Common complements: SPEC-05; TEST-03; OUT-02

SPEC-04 — Examples and counterexamples

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test	Preventive	Behavioural	All	Low	quality; misalignment

What it is: Provide representative desired examples and unacceptable counterexamples to reduce ambiguity, especially around difficult boundaries.

How to implement: Use a small curated set covering common and high-risk cases. Version examples with the rule they illustrate and include them in evaluation.

How to test / evidence: Test nearby variants rather than exact examples to verify the behaviour generalises instead of being memorised.

Limitations / what it does not protect against: Examples can bias behaviour too narrowly or create brittle imitation rather than general understanding.

Common complements: SPEC-03; TEST-02; TEST-06

SPEC-05 — Structured input contracts

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Deterministic	All	Medium	security; quality; unauthorised action

What it is: Constrain important inputs into typed fields, schemas or forms rather than relying exclusively on free text. Structure separates data from instruction and exposes missing parameters.

How to implement: Define required fields, types, ranges, enumerations and provenance. Validate before the model or tool receives the data, especially for authority-sensitive values.

How to test / evidence: Fuzz invalid, missing, oversized and malicious field values. Verify rejection or safe handling before downstream action.

Limitations / what it does not protect against: Not all tasks can be fully structured, and free-text fields can still carry malicious content.

Common complements: AUTH-07; OUT-01; SEC-01

SPEC-06 — Ambiguity and uncertainty escalation rules

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Preventive	Behavioural	All	Low	hallucination; misalignment; unauthorised action

What it is: Define when the agent must ask, abstain or escalate rather than choose an interpretation silently. This control is the first of three that together handle a task the agent should not simply answer: SPEC-06 sets the trigger, BEH-05 makes declining an acceptable outcome, and HUM-03 provides the destination that receives it. This card defines the trigger.

How to implement: Use triggers such as missing mandatory data, multiple plausible interpretations, conflicting sources, low confidence or actions above a consequence threshold. Make escalation a valid success path.

How to test / evidence: Create underspecified tasks and measure whether the agent seeks clarification at the intended threshold without over-escalating routine work.

Limitations / what it does not protect against: Model self-confidence can be poorly calibrated and excessive escalation can make the system unusable.

Common complements: BEH-05; BEH-06; HUM-03

SPEC-07 — Instruction and policy versioning / traceability

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate; Change	Preventive; Detective	Enforced	All	Medium	accountability; change; auditability

What it is: Version instructions, policies, examples and referenced knowledge that materially influence behaviour so decisions can be reconstructed and changes evaluated.

How to implement: Store immutable versions or hashes with release metadata. Record which instruction set was active for material runs and link changes to approval and regression evidence.

How to test / evidence: Reconstruct a historical decision and identify the exact instruction and policy versions. Test rollback to a prior known-good set.

Limitations / what it does not protect against: Versioning proves traceability, not correctness.

Common complements: CHG-01; OBS-02; GOV-03

Architecture & workflow

Places critical controls in system design: deterministic boundaries, explicit state, separation of duties and reversible execution.

ARCH-01 — Deterministic-versus-agentic boundary

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build	Preventive	Deterministic	All	Medium	unauthorised action; reliability; quality

What it is: Decide which decisions genuinely need model judgement and which should remain deterministic software, rules or workflow. Hard constraints are usually safer outside the model.

How to implement: For each step ask whether code, a policy engine, a query or fixed workflow can specify it reliably. Keep invariants and irreversible permission decisions deterministic unless justified.

How to test / evidence: Replace the agent with adversarial or nonsensical output and verify deterministic boundaries still prevent prohibited outcomes.

Limitations / what it does not protect against: Excessive determinism can remove useful flexibility and create brittle rule systems.

Common complements: BEH-04; OUT-02; PUR-03

ARCH-02 — Explicit workflow / state machine

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Operate	Preventive	Deterministic	Medium+	High	process integrity; unauthorised action; reliability

What it is: Represent permitted states and transitions explicitly so the agent cannot invent process stages or skip mandatory controls.

How to implement: Use a state machine or orchestrator for high-consequence processes. Let the model recommend transitions, but validate whether they are permitted and prerequisites exist.

How to test / evidence: Attempt to jump states, repeat completed states and enter impossible transitions. Verify rejection by the orchestrator.

Limitations / what it does not protect against: State machines can be expensive for open-ended work and complex when every exception is encoded.

Common complements: RUN-05; HUM-02; OBS-02

ARCH-03 — Planner-executor separation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Operate	Preventive	Enforced	Medium+	Medium	unauthorised action; misalignment; security

What it is: Separate planning from action execution so plans can be inspected, validated or constrained before tools are invoked.

How to implement: Use different components or permission scopes. The planner should not automatically inherit executor credentials. Apply policy to plans or each proposed action.

How to test / evidence: Feed a malicious or overbroad plan into the executor and verify it cannot act outside its independent permission and policy envelope.

Limitations / what it does not protect against: A bad plan can still influence a permissive executor. Separation only helps when controls are genuinely independent.

Common complements: AUTH-01; BEH-04; RUN-05

ARCH-04 — Bounded task decomposition and delegation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Preventive	Enforced	Medium+	Medium	multi-agent; cost; unauthorised action; misalignment

What it is: Limit how an agent splits work or delegates it so the original objective, authority and constraints remain attached to every subtask.

How to implement: Set maximum delegation depth, task count and allowed targets. Pass a structured task envelope containing purpose, permissions, constraints and expiry.

How to test / evidence: Create recursive delegation and scope-broadening tasks. Verify depth limits, inherited constraints and termination behaviour.

Limitations / what it does not protect against: Tight bounds can reduce performance on genuinely complex work and add orchestration overhead.

Common complements: MULTI-03; RUN-01; RUN-02

ARCH-05 — Generation-verification separation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Detective	Enforced	All	Medium	quality; hallucination; accountability

What it is: Do not let the same unchecked generation path be the sole judge of its own correctness. Separate production from validation using rules, tools, models or humans.

How to implement: Choose the verifier for the failure mode: tests for code, reconciliation for numbers, source checks for facts, policy engines for rules, independent agents for reasoning and humans for judgement.

How to test / evidence: Seed known defects and measure verifier detection and false positives. Test correlated failures where generator and checker share the same blind spot.

Limitations / what it does not protect against: Independent model reviewers can share the same errors. LLM-as-judge is evidence, not proof.

Common complements: OUT-02; OUT-03; MULTI-05; HUM-01

ARCH-06 — Safe defaults and fail-closed architecture

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Preventive; Recovery	Deterministic	Medium+	Medium	security; reliability; unauthorised action

What it is: When required context, policy, identity or validation is unavailable, default to the least harmful state rather than continuing optimistically.

How to implement: Define behaviour for missing permissions, unavailable policy services, malformed outputs and uncertain identity. High-consequence actions should deny, queue or escalate instead of bypassing control.

How to test / evidence: Simulate dependency outages and malformed responses. Verify controls fail in the intended direction and failures are visible.

Limitations / what it does not protect against: Fail-closed behaviour can harm availability. Low-risk cases may appropriately fail open with monitoring.

Common complements: FAIL-02; SEC-06; RUN-05

ARCH-07 — Reversible and staged execution design

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive; Recovery	Enforced	Medium+	High	unauthorised action; resilience; safety

What it is: Design actions so they can be previewed, staged, rolled back or compensated before the agent performs irreversible changes.

How to implement: Prefer draft-before-send, propose-before-commit, sandbox-before-production and staged execution. Build compensating actions where true rollback is impossible.

How to test / evidence: Inject incorrect actions and verify they can be intercepted before commit or reversed without unacceptable side effects.

Limitations / what it does not protect against: Some actions are inherently irreversible or externally visible; staging cannot eliminate all consequence.

Common complements: HUM-02; FAIL-03; RUN-05

Context & data

Controls what the agent may know, retrieve, remember, infer and expose, and the provenance and quality of that context.

CTX-01 — Context minimisation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Operate	Preventive	Enforced	All	Medium	privacy; prompt injection; quality; cost

What it is: Provide only the information needed for the current task, reducing distraction, leakage and opportunities for malicious or stale content to influence behaviour.

How to implement: Retrieve context just in time; avoid dumping entire mailboxes, repositories or knowledge bases into prompts. Separate mandatory instructions from task data and trim irrelevant history.

How to test / evidence: Compare performance and leakage with minimal versus oversized context. Insert malicious irrelevant content and verify it does not enter the decision path.

Limitations / what it does not protect against: Too little context can reduce accuracy or increase retrieval cost. The right window is task-specific.

Common complements: CTX-02; CTX-03; SEC-01

CTX-02 — Trusted-source and provenance controls

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	Medium	prompt injection; hallucination; compliance

What it is: Classify context sources by trust and preserve provenance so the system can distinguish authoritative policy from untrusted external content.

How to implement: Maintain authoritative-source allowlists, attach source identifiers and trust metadata, and prevent untrusted content from masquerading as system instruction.

How to test / evidence: Introduce conflicting facts from trusted and untrusted sources. Confirm intended precedence and provenance recording.

Limitations / what it does not protect against: Authoritative sources can still be wrong or compromised. Provenance is not truth.

Common complements: SPEC-02; OUT-03; SEC-03

CTX-03 — Grounded retrieval with source citation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	Medium	hallucination; quality; auditability

What it is: Require material factual claims or decisions to be grounded in retrievable evidence when authoritative sources exist.

How to implement: Use retrieval with source IDs, snippets or locations. Require evidence references for high-impact conclusions and expose them to reviewers and validators.

How to test / evidence: Use cases where model memory conflicts with the source of truth. Verify citations support the conclusion and missing evidence triggers abstention or escalation.

Limitations / what it does not protect against: Citations can be fabricated, irrelevant or misinterpreted. Grounding improves verifiability but not certainty.

Common complements: OUT-03; BEH-05; OBS-02

CTX-04 — Memory scope, retention and expiry

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate; Retire	Preventive	Enforced	All	Medium	privacy; security; drift

What it is: Limit what the agent may remember, for whom, for how long and for what purpose. Persistent memory creates both capability and risk.

How to implement: Separate session, user, project and organisational memory; set retention and expiry; prevent sensitive transient content from becoming long-term memory by default.

How to test / evidence: Verify memory is unavailable outside intended scope, expires correctly, and remains deleted across caches or derived summaries.

Limitations / what it does not protect against: Short retention reduces continuity; long retention increases privacy, contamination and stale-context risk.

Common complements: CTX-05; CTX-07; GOV-02

CTX-05 — Memory write validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Enforced	Medium+	Medium	memory poisoning; privacy; quality

What it is: Validate information before it becomes persistent memory so malicious, false or accidental content does not poison future decisions.

How to implement: Define what may be written automatically versus requiring evidence or confirmation. Store provenance, writer and timestamp; support correction and deletion.

How to test / evidence: Attempt to insert false facts, malicious instructions and another user's data. Verify rejection, quarantine or review.

Limitations / what it does not protect against: Truth can be difficult to establish at write time; validated memory can still become stale.

Common complements: CTX-04; CTX-06; SEC-01

CTX-06 — Freshness and staleness controls

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	Low	quality; reliability; compliance

What it is: Ensure time-sensitive context is current enough for the decision and visibly mark stale or undated information.

How to implement: Attach timestamps and freshness requirements. Re-query dynamic data before consequential actions and define maximum age by information type.

How to test / evidence: Serve stale prices, permissions, policy versions or schedules and verify the system refreshes or warns rather than acting silently.

Limitations / what it does not protect against: Fresh data can still be inaccurate; refreshing adds latency and cost.

Common complements: CTX-02; OUT-03; OBS-05

CTX-07 — Context, data and tenant isolation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Operate	Preventive	Deterministic	All	High	privacy; security; unauthorised action

What it is: Prevent information, memory, tools or state from one user, customer, task or environment leaking into another.

How to implement: Use tenant-aware storage, scoped retrieval, separate credentials, environment boundaries and explicit context reset. Treat traces and memory as controlled data stores.

How to test / evidence: Run cross-tenant and cross-session tests attempting to retrieve, infer or act on another party's data, including concurrent and cached-state cases.

Limitations / what it does not protect against: Isolation can fail through shared caches, logs, embeddings or misconfigured retrieval layers.

Common complements: AUTH-03; SEC-04; OBS-01

CTX-08 — Personal data in memory and traces

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate; Retire	Preventive	Procedural	All	Medium	privacy; compliance; accountability

What it is: Establish whether the agent's memory, retrieved context and execution traces contain personal data, and reconcile the retention they require with the expiry that memory scoping imposes. An agent's observability controls and its memory controls pull in opposite directions, and personal data is the material both are holding.

How to implement: Determine which stores are in scope: memory, retrieved context retained in traces, tool inputs and outputs, and the decision record. Where CTX-04 sets an expiry and OBS-01/OBS-02 set a retention, resolve which governs and record the resolution — the common outcome is that the trace outlives the memory it was derived from, and that should be a decision rather than a consequence. Point at the organisation's data-protection regime for lawful basis, rights handling and schedules; do not restate them here.

How to test / evidence: Take a subject identifiable in agent memory, apply the memory expiry, and then search the traces and decision record for the same subject. Whatever remains is what the agent actually retains. Evidence is that search, dated, with the resolution decision beside it.

Limitations / what it does not protect against: It does not establish a lawful basis and is not a substitute for a data-protection assessment. Copies exported to downstream systems are outside it (CHG-05).

Common complements: CTX-04; OBS-01; OBS-02; CHG-05

CTX-09 — Quality of owned datasets

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Operate	Preventive; Detective	Procedural	All	Medium	quality; reliability; misalignment; accountability

What it is: Govern the datasets the organisation itself builds and controls — retrieval corpora, evaluation and golden sets, few-shot exemplars and labelled outcome data — for completeness, accuracy, representativeness and labelling quality. Trust, grounding and freshness controls establish where a dataset came from, whether claims drawn from it are supported, and whether it is current. None of them establishes whether it is correct.

How to implement: Name an owner for each owned dataset. State what each is representative of, and against what population — an evaluation set that does not represent the deployed distribution measures the wrong thing accurately. Define acceptance criteria before the dataset is used rather than after it disappoints, and record labelling provenance where labels were produced by a model, by a person, or by both. Treat dataset change as a change event under CHG-01–CHG-04.

How to test / evidence: Sample and re-label independently, and report agreement rather than asserting quality. For an evaluation set, compare its distribution to production traffic and state the divergence. Evidence is the owner, the acceptance criteria, the sampling result and the stated representativeness claim — including the population it does not cover.

Limitations / what it does not protect against: It does not reach the foundation model's training data, which the organisation neither sees nor controls. A dataset can meet every criterion and still encode the assumptions of the people who built it.

Common complements: CTX-02; CTX-03; TEST-02; CHG-04

Tools, permissions & authority

Constrains what the agent can access or change, under whose identity, for how long and within what transactional limits.

AUTH-01 — Least-privilege authority

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Deterministic	All	Medium	unauthorised action; security; privacy

What it is: Give the agent only the permissions necessary for its role and current task, reducing the blast radius of mistakes or compromise.

How to implement: Define permissions from required actions rather than copying a human user's broad access. Separate read, write, approve, administer and delegate rights; revisit them as the role changes.

How to test / evidence: Attempt actions just outside task scope and confirm denial at the system boundary, not merely model refusal.

Limitations / what it does not protect against: Least privilege can create operational friction and permission sprawl if roles are poorly designed.

Common complements: PUR-05; AUTH-03; OBS-02

AUTH-02 — Tool allowlisting and capability restriction

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive	Deterministic	All	Low	tool misuse; unauthorised action; security

What it is: Restrict which tools or functions an agent can invoke rather than exposing every available capability.

How to implement: Register tools explicitly by role and use case. Remove dangerous general-purpose tools when a narrow tool will do and default new tools to unavailable until reviewed.

How to test / evidence: Try to invoke unlisted tools directly, through aliases and through another agent. Verify enforcement occurs outside the model.

Limitations / what it does not protect against: A trusted tool may still be misused with dangerous parameters; allowlisting must be paired with parameter and resource controls.

Common complements: AUTH-06; AUTH-07; SEC-03

AUTH-03 — Scoped credentials and agent identity

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive; Detective	Deterministic	All	Medium	security; auditability; unauthorised action

What it is: Use credentials tied to the agent's identity and purpose rather than shared human or administrator credentials. This makes authority enforceable and actions attributable.

How to implement: Issue service identities with narrow roles, audience and environment scope. Avoid long-lived shared secrets and separate development, test and production identities.

How to test / evidence: Inspect access logs and verify actions are attributable to the correct agent identity. Attempt credential reuse across tools, tenants or environments.

Limitations / what it does not protect against: Identity proves which credential acted, not whether the reasoning was legitimate. That limitation has a sharper form once other systems trust this identity as a principal: a credential that is used by something other than the agent produces a trail that is correct and wrong at the same time — the right identity, the wrong actor — and nothing in the record distinguishes them. Where the consequence warrants it, bind the identity to a channel that can be verified independently of the credential. Where the agent authenticates as a person rather than under its own identity, attribution collapses to that person, and this control and the permission controls that depend on it degrade to advisory.

Common complements: AUTH-04; MULTI-02; OBS-02

AUTH-04 — Just-in-time / ephemeral authorisation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive	Deterministic	Medium+	High	security; unauthorised action

What it is: Grant sensitive authority only when needed and for a bounded time or transaction, reducing standing privilege.

How to implement: Use short-lived tokens, approval-triggered elevation or per-action capability grants. Bind the grant to purpose, resource, amount and expiry where possible.

How to test / evidence: Attempt reuse after expiry, for a different resource or after the approved action completes. Confirm automatic revocation.

Limitations / what it does not protect against: JIT adds latency and identity-infrastructure dependency. Poor caching can recreate standing privilege.

Common complements: AUTH-03; HUM-02; OBS-02

AUTH-05 — Resource and destination allowlists

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive	Deterministic	All	Medium	data exfiltration; unauthorised action; security

What it is: Constrain where the agent may read, write, send or connect, even when it possesses a general tool capability.

How to implement: Allowlist repositories, folders, databases, domains, recipients, accounts or endpoints. For dynamic destinations, apply policy and approval rather than unrestricted access.

How to test / evidence: Attempt actions against similarly named, newly created and external resources. Verify canonical identity checks rather than fragile string matching.

Limitations / what it does not protect against: Static allowlists may not scale to dynamic environments and can become stale. An allowed destination can be used as a proxy or relay; name-based lists do not constrain the effective target.

Common complements: SEC-04; AUTH-02; AUTH-07

AUTH-06 — Action and transaction limits

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Deterministic	Medium+	Medium	financial loss; abuse; cost; unauthorised action

What it is: Cap the magnitude, frequency or cumulative impact of agent actions so a single mistake cannot create unlimited harm.

How to implement: Set per-action and rolling limits for money, records changed, messages sent, files deleted, compute spent or other consequential units. Trigger approval or pause near thresholds.

How to test / evidence: Attempt one oversized action and many small actions that aggregate beyond the cap. Confirm limits cannot be bypassed through retries or delegation.

Limitations / what it does not protect against: Limits reduce blast radius but do not guarantee individual actions are correct. Attackers may remain below thresholds.

Common complements: RUN-04; HUM-02; MULTI-03

AUTH-07 — Tool parameter validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Deterministic	All	Medium	tool misuse; security; quality

What it is: Validate tool arguments against schemas, ranges, policy and business rules before execution. A permitted tool call can still be dangerous if its parameters are wrong.

How to implement: Use typed functions, canonical identifiers, validation libraries and policy checks. Reject unexpected fields and normalise inputs before policy evaluation.

How to test / evidence: Fuzz boundary values, encodings, injection payloads and contradictory fields. Verify that known invalid parameters are rejected before reaching the downstream system, across the fuzzed corpus.

Limitations / what it does not protect against: Valid parameters can still represent a harmful intent; parameter validation is necessary but not sufficient.

Common complements: SPEC-05; OUT-02; SEC-01

AUTH-08 — Read/write and maker-checker separation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Deterministic	Medium+	High	unauthorised action; fraud; accountability

What it is: Separate authority to analyse or propose from authority to commit, approve or execute, particularly for high-consequence changes.

How to implement: Use read-only analyst agents, draft-producing agents and distinct approval/execution roles or identities. Do not let an agent self-approve its own high-risk action.

How to test / evidence: Attempt to move directly from analysis to commit using the same identity or delegated agent. Confirm a distinct boundary is required.

Limitations / what it does not protect against: Separation adds latency and may be unnecessary for reversible low-risk actions. Correlated automation can still defeat weak maker-checker designs.

Common complements: ARCH-05; HUM-02; MULTI-05

Behavioural & policy

Shapes model behaviour where deterministic enforcement is not practical, including policy reasoning, abstention and disclosure.

BEH-01 — Behavioural policy / constitution

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Behavioural	All	Low	misalignment; safety; compliance

What it is: Provide clear behavioural principles and policy constraints the agent should apply when interpreting ambiguous tasks.

How to implement: Keep principles concise, prioritised and tied to examples. Reference authoritative policies and state conflict handling. Use this as guidance, not a substitute for hard enforcement.

How to test / evidence: Construct dilemmas where objectives conflict with policy principles and evaluate consistency across repeated trials.

Limitations / what it does not protect against: Natural-language policies are probabilistic and may be overridden by context or attack.

Common complements: SPEC-02; BEH-04; TEST-07

BEH-02 — Scope and relevance guardrail

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Enforced	All	Medium	scope creep; prompt injection; misuse

What it is: Detect requests or intermediate plans outside the agent's defined purpose and redirect, refuse or escalate them.

How to implement: Combine lightweight classifiers, rules and model-based checks. Run on user input and, for autonomous systems, on proposed subgoals and actions.

How to test / evidence: Test clear, borderline and disguised out-of-scope requests. Measure false accepts and false rejects by risk tier.

Limitations / what it does not protect against: Classifiers can be bypassed and may block legitimate adjacent work; boundaries shift over time.

Common complements: PUR-01; TEST-06; OBS-04

BEH-03 — Safety and content guardrails

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	Medium	safety; privacy; compliance; abuse

What it is: Filter or classify content that violates safety, legal, privacy or organisational policy before it is accepted, generated or acted upon.

How to implement: Use specialised classifiers, deterministic patterns and domain rules. Apply at input and output where risks differ, and log high-risk trips for review.

How to test / evidence: Use representative violations and adversarial paraphrases. Measure false negatives, false positives and bypass through tools or encoded content.

Limitations / what it does not protect against: Content guardrails do not control all agent actions and can be brittle outside their trained domains.

Common complements: SEC-01; OUT-02; OBS-04

BEH-04 — External deterministic policy engine

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive	Deterministic	Medium+	High	unauthorised action; compliance; security

What it is: Enforce hard organisational rules outside the model using a policy engine or deterministic service that can allow, deny or require approval for proposed actions.

How to implement: Express policies over identity, action, resource, context, risk and thresholds. Evaluate immediately before execution and record the policy decision and version.

How to test / evidence: Generate proposals that violate policy in subtle ways and verify the engine blocks them regardless of model explanation. Test policy-service outage behaviour.

Limitations / what it does not protect against: Policy engines only enforce encoded rules supplied with correct context. Complex rules can become difficult to maintain.

Common complements: AUTH-07; ARCH-06; OBS-02

BEH-05 — Abstention / refusal behaviour

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Behavioural	All	Low	hallucination; safety; unauthorised action

What it is: Make "I cannot safely determine or perform this" an acceptable outcome when evidence, authority or confidence is insufficient. This control is the second of three that together handle a task the agent should not simply answer: SPEC-06 sets the trigger, BEH-05 makes declining an acceptable outcome, and HUM-03 provides the destination that receives it. This card supplies the disposition — a trigger with no acceptable way to decline produces an agent that proceeds anyway.

How to implement: Define refusal reasons and recovery paths: ask for data, hand over, route to another system or log an exception. Avoid rewarding agents solely for task completion.

How to test / evidence: Create cases with missing evidence, prohibited action or low confidence and measure appropriate abstention versus unnecessary refusal.

Limitations / what it does not protect against: Models may over-refuse or confidently proceed despite uncertainty. Refusal policy needs calibration and alternatives.

Common complements: SPEC-06; BEH-06; HUM-03

BEH-06 — Uncertainty and confidence handling

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Operate	Preventive; Detective	Enforced	All	Medium	quality; hallucination; safety

What it is: Represent uncertainty explicitly and tie it to behaviour such as extra verification, reduced authority or escalation.

How to implement: Use observable signals rather than only self-reported confidence: source disagreement, missing fields, verifier scores, out-of-distribution signals and repeated-run variance. Calibrate thresholds on real data.

How to test / evidence: Measure calibration and test whether lower confidence changes the action path as intended.

Limitations / what it does not protect against: LLM confidence statements are often poorly calibrated and can be manipulated. Composite signals are better but imperfect.

Common complements: TEST-07; OUT-03; HUM-03

BEH-07 — Disclosure of AI interaction

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Operate	Preventive	Enforced	All	Low	compliance; accountability; misuse

What it is: Ensure a person interacting with the agent is informed they are interacting with an AI system, in every channel the agent can reach them through — not only the one it was designed around.

How to implement: Attach the disclosure to the agent's identity rather than to a single interface, so it travels with any channel the agent acquires. Enumerate the outbound channels: chat, email, ticketing, documents, voice, API responses consumed by another party's interface. Decide the disclosure form per channel and record which channels are in scope. Where the agent delegates to a sub-agent or a tool that communicates on its behalf, the obligation follows the interaction, not the component.

How to test / evidence: Exercise each enumerated outbound channel and confirm the disclosure is present and legible at first contact. Add a channel in test and confirm the review catches that it has no disclosure decision. Evidence is the channel inventory with a disclosure decision against each, and the exercised output.

Limitations / what it does not protect against: Disclosure is not consent and is not a substitute for a legal basis. It does not reach channels the agent acquires without review, which is why the inventory is the control rather than the wording. Disclosure also addresses the agent that is not identified as an AI system; it does not address one that actively presents as a named person — a persona, a signature block, or a sender identity belonging to a human. Where the agent can do that, it is a separate decision with its own approval, and the record should state which human identities the agent may and may not speak as.

Common complements: PUR-01; BEH-06; HUM-04; AUTH-05

BEH-08 — Reasons for a decision

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Operate	Preventive; Detective	Procedural	Medium+	Medium	accountability; compliance; quality

What it is: Produce an account of why the agent decided as it did that is intelligible to the audience entitled to it, and faithful to what actually determined the outcome. Retrievable evidence and a durable audit trail are inputs to this and are not substitutes for it: evidence shows what was available, a trail shows what happened, and neither shows why.

How to implement: Identify the audiences and what each is owed — an affected person, an internal reviewer, an auditor and a regulator need different accounts of the same decision, and only one of them can read a trace. Derive the explanation from the recorded decision path rather than generating it afterwards, so that the account and the trail cannot diverge. Where the agent cannot produce a faithful account, that is itself the finding: record it and constrain the decision's autonomy accordingly.

How to test / evidence: Take a sample of consequential decisions and have someone from each audience read the explanation without access to the trace. Test faithfulness separately: change a determining input, and confirm the explanation changes with it. An explanation that does not move when the inputs move is a description of the system, not of the decision.

Limitations / what it does not protect against: A fluent explanation can be unfaithful, and fluency is what this control most easily produces. Faithfulness testing bounds that and does not eliminate it. Explanation is not justification: an intelligible account of a wrong decision remains a wrong decision.

Common complements: CTX-03; OBS-02; HUM-01; BEH-06

Runtime execution

Bounds execution time, cost, retries, loops, actions and concurrency while the agent is operating.

RUN-01 — Step, loop and action limits

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive	Deterministic	Medium+	Low	runaway behaviour; cost; reliability

What it is: Limit reasoning cycles, tool calls, subtasks or actions an agent may perform without renewal or escalation.

How to implement: Set task-specific ceilings and detect repeated states or no-progress loops. Require a checkpoint or human review before extending high-impact runs.

How to test / evidence: Create cyclic tasks, failing tools and recursive delegation. Confirm predictable termination and a diagnosable trace.

Limitations / what it does not protect against: Hard limits can terminate legitimate complex work; limits should reflect task class and risk.

Common complements: ARCH-04; RUN-02; HUM-03

RUN-02 — Time, token and cost budgets

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive	Deterministic	All	Low	cost; denial of service; reliability

What it is: Bound resources an agent may consume so errors, loops or attacks cannot create uncontrolled cost.

How to implement: Set per-run and rolling budgets for tokens, compute, paid APIs and elapsed time. Surface remaining budget to orchestration logic and degrade gracefully.

How to test / evidence: Force repeated retries or expensive branches and confirm the run stops, simplifies or escalates at the boundary.

Limitations / what it does not protect against: Budgets can cut off valuable work and do not address harmful low-cost actions.

Common complements: RUN-01; RUN-03; OBS-03

RUN-03 — Retry limits and backoff

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Recovery	Deterministic	All	Low	reliability; duplicate action; cost

What it is: Prevent repeated retries from amplifying a bad action, duplicating transactions or overwhelming dependencies.

How to implement: Define errors eligible for retry, maximum attempts, backoff and when to require human or alternative-path intervention. Pair with idempotency for writes.

How to test / evidence: Simulate transient and permanent failures. Confirm permanent errors do not loop and successful retries do not duplicate effects.

Limitations / what it does not protect against: Too-conservative retries reduce resilience; permissive retries amplify incidents.

Common complements: RUN-06; FAIL-02; OBS-04

RUN-04 — Rate and concurrency limits

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive	Deterministic	All	Medium	abuse; cost; reliability; unauthorised action

What it is: Limit how rapidly or concurrently an agent can act, reducing blast radius, overload and automation abuse.

How to implement: Apply per-agent, user, tenant, tool and destination limits. Use lower ceilings for destructive actions and distinguish reads from writes.

How to test / evidence: Launch concurrent tasks designed to exceed limits and verify atomic throttling that cannot be bypassed by multiple agents.

Limitations / what it does not protect against: Rate limits may delay legitimate bursts and do not stop low-frequency harmful actions.

Common complements: AUTH-06; SEC-06; OBS-03

RUN-05 — Runtime checkpoints / pre-action gates

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Enforced	Medium+	Medium	unauthorised action; stale context; reliability

What it is: Pause at defined points to revalidate state, policy, evidence or approval, especially immediately before consequential actions.

How to implement: Place gates before external communication, commit, payment, deletion, privilege change or irreversible transition. Re-check fresh context before execution.

How to test / evidence: Change relevant state between planning and execution and verify the checkpoint catches the stale assumption.

Limitations / what it does not protect against: Too many checkpoints add latency and can become ceremonial if they always pass. A check-then-act gate can be raced: state changed between the check and the action is not revalidated.

Common complements: BEH-04; HUM-02; CTX-06

RUN-06 — Idempotency and duplicate-action protection

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Deterministic	All	Medium	duplicate action; financial loss; reliability

What it is: Ensure retries, duplicate messages or repeated reasoning cannot cause the same side effect more than once.

How to implement: Use idempotency keys, transaction identifiers and deduplication stores for writes, sends and payments. Make completion state visible to the agent.

How to test / evidence: Replay identical and semantically equivalent requests under timeout conditions and confirm only one side effect occurs.

Limitations / what it does not protect against: Some actions are difficult to make idempotent and semantic duplicates may not share an identifier.

Common complements: RUN-03; OBS-02; FAIL-03

RUN-07 — Sandboxed execution environment

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Deterministic	Medium+	High	code execution; security; data exfiltration

What it is: Execute code, files or untrusted operations in an isolated environment with constrained filesystem, network, process and credential access.

How to implement: Use ephemeral containers or sandboxes, minimal images, resource limits and controlled egress. Destroy or reset state between untrusted tasks where possible.

How to test / evidence: Attempt filesystem escape, network access, secret discovery, persistence and resource exhaustion. Confirm containment at the platform layer.

Limitations / what it does not protect against: Sandboxes can have vulnerabilities and may not replicate production behaviour exactly.

Common complements: SEC-04; SEC-05; AUTH-03

Output validation

Checks structure, policy, facts, code and downstream compatibility before outputs are trusted or acted upon.

OUT-01 — Structured output schemas

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive	Deterministic	All	Low	quality; integration; unauthorised action

What it is: Require machine-consumed outputs to conform to explicit schemas, types and enumerations rather than unconstrained prose.

How to implement: Use JSON/schema validation, typed models or function-call contracts. Reject, repair or escalate malformed output before downstream use.

How to test / evidence: Fuzz missing fields, extra fields, wrong types, invalid enums and oversized values. Verify that the malformed outputs in the test corpus do not trigger action, and record the corpus, the population it samples and the period covered.

Limitations / what it does not protect against: Schema-valid output can still be semantically wrong or harmful.

Common complements: SPEC-05; OUT-02; AUTH-07

OUT-02 — Deterministic business-rule validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Deterministic	All	Medium	quality; compliance; financial loss

What it is: Check agent conclusions or proposed actions against invariant business rules before acceptance or execution.

How to implement: Encode balances, eligibility, totals, limits, required approvals and allowed transitions in deterministic validators. Return explicit failure reasons.

How to test / evidence: Create outputs that are plausible but violate one invariant at a time. Confirm detection and safe failure.

Limitations / what it does not protect against: Business rules may be incomplete or change frequently; validators need governance like any other code.

Common complements: ARCH-01; BEH-04; TEST-02

OUT-03 — Factual and source verification

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Operate	Detective	Enforced	All	Medium	hallucination; quality; compliance

What it is: Verify material factual claims against authoritative sources before they drive decisions or are communicated as fact.

How to implement: Require citations for high-impact claims, retrieve referenced sources and compare key entities, dates, amounts or assertions. Escalate unresolved contradictions.

How to test / evidence: Use known traps, outdated information and fabricated citations. Measure whether verification catches unsupported claims.

Limitations / what it does not protect against: Source verification is harder for judgement, forecasts and synthesis; authoritative sources can disagree.

Common complements: CTX-03; CTX-06; HUM-01

OUT-04 — Code and executable-artifact validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	Medium	security; reliability; quality

What it is: Treat generated code, queries, scripts, infrastructure changes and other executable artefacts as untrusted until technically validated.

How to implement: Apply linting, static analysis, unit tests, security scanning, compilation, sandbox execution and policy checks as appropriate. Require review for privileged changes.

How to test / evidence: Seed insecure dependencies, injection flaws, destructive commands and logic errors. Verify the pipeline blocks them.

Limitations / what it does not protect against: Automated tools have blind spots and may not detect intent-level design errors.

Common complements: TEST-01; SEC-05; HUM-01

OUT-05 — Downstream contract and compatibility validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy; Operate	Preventive; Detective	Deterministic	All	Medium	integration; reliability; cascading failure

What it is: Validate that outputs meet the technical and semantic contract expected by downstream systems, agents or humans.

How to implement: Test required fields, units, encodings, ordering, API versions, document templates and domain semantics. Use contract tests at boundaries.

How to test / evidence: Change downstream versions and assumptions to verify incompatibilities are detected before production impact.

Limitations / what it does not protect against: Contract tests cannot detect every semantic misunderstanding between independently evolving components.

Common complements: TEST-04; MULTI-01; CHG-04

OUT-06 — Synthetic-content marking

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Operate	Preventive; Detective	Enforced	All	Medium	compliance; misuse; accountability

What it is: Mark generated or manipulated audio, image, video and text so that it is detectable as artificially generated, in a machine-readable form, and keep the marking attached as the content moves.

How to implement: Apply the marking where the content is produced rather than where it is published, so that content leaving by an unanticipated route is still marked. Choose a form that survives the handling the content will actually receive — re-encoding, copy-paste, extraction from a document — and record what it does not survive. Where content passes through a tool that strips metadata, treat that tool as in scope for the control rather than as an exception to it.

How to test / evidence: Generate content of each type, put it through the pipeline it will really travel, and detect the marking at the far end. Then check the surfaces the content is actually published to: does the mark survive ingest, and is it displayed? Evidence is the detection result per content type per route, a stated list of routes that strip it, and a stated list of destinations that retain the mark but do not surface it — those are the ones where the control reports success and the reader learns nothing.

Limitations / what it does not protect against: Marking is removable by anyone who wants it removed; it addresses accidental circulation rather than adversarial laundering, and it does not make the content accurate. More importantly, marking succeeds at production and can still fail at the point of use. A mark that is present, cryptographically sound and correctly attached achieves nothing if the surface displaying the content does not read it, does not trust it, or strips it on ingest. The control's own report — content marked — is true, and is not evidence that any reader will see the mark. Where that matters, the measure is whether the marking survives and is surfaced on the routes the content actually travels, not whether it was applied.

Common complements: OUT-01; OBS-02; CTX-03; BEH-07

Testing & evaluation

Builds evidence through component, regression, scenario, adversarial, integration, UAT, shadow and release-gate evaluation.

TEST-01 — Component and unit tests

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Change	Detective	Enforced	All	Low	quality; reliability; change

What it is: Test deterministic components and narrowly defined agent behaviours in isolation so failures can be localised before full-system evaluation.

How to implement: Unit-test parsers, policy checks, tool wrappers and state transitions conventionally; for prompts or skills, use focused behaviour cases with stable scoring criteria.

How to test / evidence: Run automatically on every relevant change and fail the build for deterministic regressions or material behavioural threshold breaches.

Limitations / what it does not protect against: Passing components do not prove the integrated agent works; model outputs can be stochastic.

Common complements: TEST-02; TEST-04; CHG-03

TEST-02 — Golden-set regression evaluation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Change	Detective	Enforced	All	Medium	change; quality; regression

What it is: Maintain a curated set of important historical, edge and failure cases so behaviour changes are detected across model, prompt and code updates.

How to implement: Include normal cases, incidents, adversarial examples and previously fixed bugs. Version expected outcomes or scoring rubrics and compare aggregate and severe-case regressions.

How to test / evidence: Re-run on every material release and investigate statistically or operationally meaningful deterioration before deployment.

Limitations / what it does not protect against: Golden sets can overfit development and become stale; supplement them with fresh and hidden cases.

Common complements: CHG-03; TEST-07; OBS-06

TEST-03 — Scenario and end-to-end system evaluation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy	Detective	Procedural	All	High	quality; reliability; misalignment

What it is: Evaluate realistic complete tasks across planning, retrieval, tools, state and outputs rather than testing only single model responses.

How to implement: Build scenarios from real workflows, including long-horizon cases and operational constraints. Score task success, safety, policy compliance, resource use and recoverability.

How to test / evidence: Run diverse scenarios and inspect traces for cases where the final answer appears correct for the wrong reason.

Limitations / what it does not protect against: End-to-end tests are slower and harder to diagnose and may still miss production diversity.

Common complements: PUR-02; OBS-01; TEST-07

TEST-04 — Tool and integration testing

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy; Change	Detective	Enforced	All	Medium	integration; reliability; security

What it is: Test interfaces between the agent and external tools, APIs, data sources and system components, including failure and permission behaviour.

How to implement: Use test environments, mocked failures and contract tests. Verify authentication, parameter handling, retries, rate limits, partial failures and side effects.

How to test / evidence: Exercise timeouts, malformed tool responses, permission changes and version mismatches. Confirm the agent does not hallucinate success.

Limitations / what it does not protect against: Mocks may not reproduce production quirks; live integration tests can be costly or risky.

Common complements: OUT-05; RUN-03; AUTH-07

TEST-05 — Adversarial testing and red teaming

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Operate; Change	Detective	Procedural	Medium+	High	security; abuse; misalignment

What it is: Actively try to make the agent violate goals, policy, security boundaries or user expectations using malicious and unusual inputs.

How to implement: Cover prompt injection, goal hijack, tool misuse, privilege abuse, exfiltration, memory poisoning, denial of service and human trust exploitation. Use independent testers where consequence warrants it.

How to test / evidence: Track exploit reproducibility, impact and control coverage; retest remediations and convert incidents into regression cases.

Limitations / what it does not protect against: Red teaming samples an open-ended attack space and cannot prove absence of vulnerabilities.

Common complements: SEC-01; SEC-06; TEST-02

TEST-06 — Boundary, negative and misuse-case testing

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test	Detective	Procedural	All	Medium	unauthorised action; safety; compliance

What it is: Test what the agent must not do, not only what it should do. Boundaries are often where natural-language specifications fail.

How to implement: Derive negative cases from prohibited outcomes, permissions, data boundaries, scope limits and reasonably foreseeable misuse. Include near-boundary legitimate cases to measure over-blocking.

How to test / evidence: Verify prohibited actions are blocked at the strongest available layer and legitimate edge cases still work.

Limitations / what it does not protect against: Negative spaces are large and known misuse cases can distract from novel behaviour.

Common complements: PUR-03; AUTH-01; BEH-02

TEST-07 — Repeated stochastic trials

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Change	Detective	Procedural	All	Medium	stochasticity; quality; reliability

What it is: Run the same or equivalent scenario multiple times to measure variance, tail failures and sensitivity to sampling or context changes.

How to implement: Choose repeat counts proportional to consequence and expected failure frequency. Report distributions and worst-case outcomes, not only mean scores.

How to test / evidence: Compare versions with confidence intervals or practical thresholds and investigate rare severe failures even when averages improve.

Limitations / what it does not protect against: Repeated trials increase cost and still may not reveal very rare events.

Common complements: BEH-06; TEST-08; OBS-03

TEST-08 — Release performance and reliability thresholds

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy; Change	Preventive	Procedural	All	Medium	quality; safety; accountability

What it is: Define quantitative or qualitative gates that must be met before an agent version can progress to production or higher autonomy.

How to implement: Set thresholds for task success, severe-error rate, policy violations, latency, cost, security findings and escalation. Use stricter gates for higher impact.

How to test / evidence: Automate release gating where measures are reliable and require documented risk acceptance for exceptions.

Limitations / what it does not protect against: Thresholds can encourage optimisation to the test and create false precision.

Common complements: PUR-02; CHG-03; GOV-03

TEST-09 — User acceptance and domain-expert validation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy	Detective	Procedural	All	Medium	quality; usability; compliance

What it is: Have representative users and subject-matter experts assess whether the agent works in the real operating context, including tacit requirements not captured by engineering tests.

How to implement: Use realistic tasks, data and workflows; collect structured feedback on correctness, usability, escalation and trust. Include risk/compliance reviewers for relevant cases.

How to test / evidence: Record acceptance criteria, unresolved issues and explicit sign-off or risk acceptance before production.

Limitations / what it does not protect against: Users may be impressed by fluency, overlook rare risks or test only happy paths.

Common complements: HUM-04; GOV-02; TEST-03

TEST-10 — Shadow, parallel and phased rollout

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive; Detective	Procedural	All	High	deployment risk; reliability; safety

What it is: Expose the agent progressively to real conditions while limiting its ability to cause harm, creating evidence before full autonomous deployment.

How to implement: Start with offline replay, shadow recommendations, parallel comparison, limited cohorts, capped authority and gradual expansion. Define rollback criteria.

How to test / evidence: Compare agent outputs and outcomes with the baseline process; monitor disagreement, near misses and control trips before increasing scope.

Limitations / what it does not protect against: Parallel operation can be expensive and may create false reassurance if people behave differently during testing.

Common complements: ARCH-07; OBS-03; FAIL-03

Human oversight

Introduces meaningful human review, approval, escalation, intervention and production sampling where judgement is needed.

HUM-01 — Human review of material outputs

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate	Preventive; Detective	Procedural	All	Medium	quality; safety; accountability

What it is: Require a competent person to review selected outputs before they are relied upon, communicated or executed.

How to implement: Define which outputs require review, what the reviewer checks, the evidence available and the time allowed. Provide sources and traces rather than only a polished answer.

How to test / evidence: Seed known defects and measure reviewer detection, time and override rate. Test whether workload allows meaningful review.

Limitations / what it does not protect against: Humans can rubber-stamp fluent outputs, miss errors or lack the context to challenge the agent.

Common complements: HUM-04; OUT-03; OBS-02

HUM-02 — Human approval gate for high-impact actions

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Preventive	Procedural	Medium+	High	unauthorised action; safety; compliance

What it is: Require explicit human authorisation before specified consequential or irreversible actions are committed.

How to implement: Trigger on action type, risk tier, thresholds, uncertainty or exceptions. Present the proposed action, evidence, rationale and alternatives; record approver identity and decision.

How to test / evidence: Attempt to bypass approval through retries, delegation, batch actions or changed parameters. Verify approval binds to the exact action approved.

Limitations / what it does not protect against: Approval becomes weak when volumes are high, context is poor or approvers click through routinely. An approval can be obtained against a misleading summary; unless the approval is bound to the exact action executed, the record proves consent to something else.

Common complements: AUTH-04; AUTH-08; HUM-04

HUM-03 — Escalation matrix and handoff

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Preventive; Corrective	Procedural	All	Medium	safety; reliability; accountability

What it is: Define when and to whom the agent must hand control when it cannot safely proceed, including domain, operational, security and compliance routes. This control is the third of three that together handle a task the agent should not simply answer: SPEC-06 sets the trigger, BEH-05 makes declining an acceptable outcome, and HUM-03 provides the destination that receives it. This card defines the destination — by route, role and service level. A trigger and a disposition with nowhere to send the result is an agent that stops and waits.

How to implement: Map trigger types to roles, service levels and information packages. Preserve task state so the human can continue without reconstructing the interaction.

How to test / evidence: Create each escalation trigger and verify correct routing, sufficient context and safe behaviour while waiting.

Limitations / what it does not protect against: If queues are slow or unclear, agents or users may route around the control.

Common complements: SPEC-06; BEH-05; FAIL-02

HUM-04 — Meaningful oversight and automation-bias controls

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Preventive; Detective	Procedural	Medium+	Medium	human trust exploitation; accountability; quality

What it is: Design human review so the person can actually challenge the agent rather than merely confirm it.

How to implement: Show uncertainty, evidence and alternatives; avoid presenting AI output as presumptively correct; train reviewers on failure modes; require independent reasoning for selected high-risk decisions.

How to test / evidence: Measure override rates, defect detection and reviewer performance with and without agent recommendations. Investigate implausibly low challenge rates.

Limitations / what it does not protect against: Human oversight remains vulnerable to fatigue, authority bias, time pressure and skill gaps. This control assumes a reviewer competent to exercise it. Reviewer competence is a precondition of meaningful oversight rather than a component of it: a reviewer who cannot recognise the agent's failure modes cannot challenge it, however well the review interface is designed. That competence is also the subject of a standing obligation on the organisation — see the discussion of AI literacy — which this catalogue points at rather than supplies.

Common complements: HUM-01; OBS-06; TEST-09

HUM-05 — Human override, cancel and stop capability

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Corrective	Enforced	Medium+	Medium	safety; incident response; unauthorised action

What it is: Give authorised humans the practical ability to interrupt, reverse or supersede an agent during operation.

How to implement: Provide visible controls to pause runs, revoke authority, cancel queued actions and replace agent output. Define safe-stop behaviour and permissions.

How to test / evidence: Exercise the control during active and degraded runs; confirm it takes effect quickly and does not leave partial unsafe state.

Limitations / what it does not protect against: Superseding an agent's output does not reverse what that output has already caused: where an action has propagated to another system or another person, this control replaces the agent's next step, not its last one. Revoking the agent's authority stops the agent and not the delegations it has already issued, which outlive their grantor. And routine supervision has a latency floor — the control is exercised while the agent runs, so an agent that completes faster than a person can attend to it offers no window in which authority can be exercised at all. That is a property of the agent's design rather than of the control, and it belongs in the design decision rather than in the incident.

Common complements: FAIL-01; FAIL-03; OBS-04

HUM-06 — Human quality sampling and periodic review

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate; Change	Detective	Procedural	All	Medium	quality; drift; accountability

What it is: Review a risk-based or statistical sample of production work even when individual actions do not require pre-approval.

How to implement: Stratify samples by risk, version, user, outcome and anomaly. Feed findings into evals, controls and risk assessment.

How to test / evidence: Track defect trends and whether sampled findings predict later incidents. Increase sampling when drift or change occurs.

Limitations / what it does not protect against: Sampling can miss rare severe failures and can become ritual if findings do not drive action.

Common complements: OBS-05; OBS-06; TEST-02

HUM-07 — Contestability and recourse

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Corrective	Procedural	All	Medium	accountability; compliance; quality

What it is: Provide a person affected by an agent's outcome with a route to challenge it, obtain human reconsideration, and where applicable opt out — and an answer. Capturing a complaint as evidence is not recourse: it improves the system and owes the complainant nothing.

How to implement: Define what is contestable, by whom, and within what period. Identify the contestable unit before deployment: where an outcome emerges from a sequence, decide whether the person contests the outcome, the action or the decision to act, and publish that. Where the agent can act irreversibly, either stage the action so a contest window exists (ARCH-07) or state that no such window does. Route contests to a human with the authority to reverse the outcome, not only to review it.

How to test / evidence: Submit a contest as an affected person would, through the published route, and follow it to an outcome. Evidence is the response, its timeliness against the stated period, and whether reversal was actually available. A route that terminates in an internal ticket is a captured complaint, not a recourse mechanism, and the test should distinguish them.

Limitations / what it does not protect against: Recourse is not accuracy — an available route does not make outcomes right, and a well-run process can uphold a wrong decision. It cannot reverse effects already propagated to third parties.

Common complements: OBS-06; HUM-03; ARCH-07; BEH-08

Multi-agent systems

Controls communication, identity, delegation, orchestration, independent challenge and conflict among interacting agents.

MULTI-01 — Explicit inter-agent communication contract

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Test; Operate	Preventive	Enforced	Medium+	Medium	multi-agent; prompt injection; integration

What it is: Define message schemas, allowed message types, semantics, trust assumptions and error handling between agents instead of treating free-form language as a trusted protocol.

How to implement: Use typed envelopes with sender, task, authority, provenance, correlation ID and expiry. Separate instruction from data and reject malformed or unauthorised message types.

How to test / evidence: Fuzz messages, omit fields, spoof roles and inject instructions into data fields. Verify receiving agents validate before acting.

Limitations / what it does not protect against: Semantic ambiguity can remain inside structured messages; contracts must evolve with agents.

Common complements: OUT-05; MULTI-02; SEC-01

MULTI-02 — Agent identity and authentication

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive	Deterministic	Medium+	Medium	identity abuse; security; multi-agent

What it is: Authenticate the agent or service sending a message or requesting action so components do not trust arbitrary claimed roles.

How to implement: Use cryptographic or platform identity, bind identity to role and permissions, and log delegation chains.

How to test / evidence: Spoof agent names, replay old messages and use a valid identity outside its allowed role. Confirm rejection.

Limitations / what it does not protect against: Authenticated agents can still be compromised or malicious; identity is only one trust signal.

Common complements: AUTH-03; MULTI-03; OBS-02

MULTI-03 — Delegated-authority inheritance and attenuation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Deterministic	High	High	privilege abuse; unauthorised action; multi-agent

What it is: Ensure delegated agents receive no more authority than the delegating agent and preferably only the subset needed for the subtask.

How to implement: Pass platform-enforced capability scopes with delegation depth and expiry. Do not let downstream agents reacquire broad credentials independently.

How to test / evidence: Attempt privilege expansion through chains of agents and verify effective permissions only decrease or remain bounded.

Limitations / what it does not protect against: Complex delegation graphs are hard to reason about and may create hidden privilege combinations.

Common complements: AUTH-01; AUTH-04; MULTI-02

MULTI-04 — Supervisor / orchestrator control

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Preventive; Detective	Enforced	High	High	coordination; runaway behaviour; reliability

What it is: Use an orchestrator to allocate work, maintain global state, enforce workflow and stop agents that diverge from the objective or resource envelope.

How to implement: Keep orchestration policy and critical state outside individual agents. Give the supervisor visibility into tasks, budgets, permissions, progress and exceptions.

How to test / evidence: Make a worker return misleading status, repeat work or exceed scope and verify the supervisor detects or constrains it.

Limitations / what it does not protect against: The supervisor becomes a high-value control point and possible single point of failure; it can share model blind spots.

Common complements: ARCH-04; RUN-01; OBS-01

MULTI-05 — Independent critic / checker agent

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Operate	Detective	Behavioural	Medium+	Medium	quality; misalignment; assurance

What it is: Use a separate agent or model to challenge plans, outputs or compliance before progression, introducing a distinct reasoning path.

How to implement: Give the checker a different role, prompt and preferably evidence view; prevent it from merely echoing the producer. Use deterministic or human checks for hard boundaries.

How to test / evidence: Seed subtle errors and measure checker catch rate and correlated misses. Compare periodically with human adjudication.

Limitations / what it does not protect against: Two agents can confidently agree on the same error, especially when they share model, data or context.

Common complements: ARCH-05; HUM-01; OUT-02

MULTI-06 — Conflict, deadlock and consensus rules

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Preventive; Corrective	Enforced	High	Medium	coordination; reliability; cost

What it is: Define how agents resolve disagreement, circular dependencies, contradictory recommendations and stalled coordination.

How to implement: Set explicit precedence, quorum, tie-break, timeout and escalation rules. Avoid indefinite debate loops and distinguish consensus from correctness.

How to test / evidence: Create conflicting outputs and circular task dependencies. Verify deterministic resolution or escalation within bounded time.

Limitations / what it does not protect against: Consensus can amplify shared bias and may hide minority but correct warnings.

Common complements: RUN-01; HUM-03; MULTI-04

Observability & monitoring

Makes behaviour, decisions, drift, incidents and the health of controls visible in production.

OBS-01 — End-to-end tracing and logging

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Detective	Enforced	All	Medium	auditability; incident response; reliability

What it is: Capture the sequence of model calls, tool calls, retrieved context, state transitions and errors needed to understand what an agent did.

How to implement: Use correlation IDs across agents and tools; record timestamps, versions and key inputs/outputs subject to privacy minimisation. Protect logs against tampering and inappropriate access.

How to test / evidence: Select a production run and reconstruct its path end to end. Confirm redaction does not prevent investigation of material actions.

Limitations / what it does not protect against: Excessive logging creates privacy, security and cost risks; hidden model reasoning may not be available or appropriate to retain.

Common complements: OBS-02; SEC-02; GOV-03

OBS-02 — Decision and action audit trail

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate	Detective	Procedural	Medium+	Medium	accountability; compliance; security

What it is: Maintain a durable record of consequential decisions and actions: who or what acted, under which authority, using which evidence and control decisions.

How to implement: Record agent identity, delegator, action, target, parameters, approval, policy decision, evidence references, version and outcome. Make records queryable. Make the record tamper-evident wherever the agent has write access to the systems that hold it. An audit trail whose integrity depends on the good behaviour of the thing it audits is not evidence — and unlike a human user, an agent is routinely granted write access to its own logging estate because that estate is just another system it integrates with. Append-only storage, write separation from the agent's own identity, or a periodic digest held outside its reach are the usual forms.

How to test / evidence: Sample material actions and verify an independent reviewer can understand and attribute the event.

Limitations / what it does not protect against: An audit trail documents bad decisions but does not prevent them; sensitive log data must be controlled.

Common complements: AUTH-03; SPEC-07; GOV-03

OBS-03 — Operational metrics and service-level objectives

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Detective	Enforced	All	Medium	reliability; quality; cost

What it is: Monitor measures that indicate whether the agent performs acceptably in production, including outcomes, latency, cost, failure, escalation and safety events.

How to implement: Define SLOs by use case and risk; separate business success from control-health metrics. Segment by version and workflow so averages do not hide local failure.

How to test / evidence: Trigger synthetic or controlled failures and confirm metrics and alerts respond. Review whether thresholds predict user impact.

Limitations / what it does not protect against: Metrics can be incomplete, delayed or gamed; qualitative harms may not fit simple telemetry.

Common complements: PUR-02; OBS-04; TEST-08

OBS-04 — Anomaly and unusual-behaviour detection

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate	Detective	Enforced	Medium+	High	security; abuse; drift; incident response

What it is: Detect deviations such as unusual tools, destinations, action volumes, failure patterns or control trips that may signal compromise or drift.

How to implement: Build baselines by agent role and risk. Alert on high-impact anomalies and aggregate lower-risk patterns for review. Include behaviour across agent chains.

How to test / evidence: Simulate atypical actions and verify detection without unsustainable false-positive volume.

Limitations / what it does not protect against: Novel legitimate work can look anomalous; sophisticated misuse may imitate normal patterns.

Common complements: SEC-06; FAIL-01; OBS-03

OBS-05 — Behaviour, model and data drift monitoring

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate; Change	Detective	Procedural	All	Medium	drift; quality; change

What it is: Detect material changes in performance or behaviour over time due to model updates, data shifts, context changes, user behaviour or environment.

How to implement: Track key evals and production metrics by version and time; periodically replay benchmark cases and monitor input distributions and escalation patterns.

How to test / evidence: Introduce known distribution shifts or compare before/after model updates to verify detection and review triggers.

Limitations / what it does not protect against: Some drift is beneficial and not all harmful change is statistically obvious.

Common complements: CHG-04; TEST-02; HUM-06

OBS-06 — User feedback, complaint and near-miss capture

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate; Change	Detective; Corrective	Procedural	All	Low	quality; safety; accountability

What it is: Treat user corrections, complaints, overrides and near misses as control evidence and a source of new tests.

How to implement: Provide low-friction reporting linked to the specific run. Classify severity, root cause and control failure; feed validated cases into regression and risk review.

How to test / evidence: Trace sampled feedback through triage, remediation and regression inclusion. Verify severe reports trigger escalation.

Limitations / what it does not protect against: Feedback is biased toward visible failures and may underrepresent harmed non-users or silent errors.

Common complements: TEST-02; HUM-06; GOV-02

OBS-07 — Control-health monitoring

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Detective	Enforced	Medium+	High	control failure; security; reliability

What it is: Monitor whether critical controls themselves are available and functioning rather than assuming their presence guarantees protection.

How to implement: Emit telemetry for policy-engine errors, guardrail bypass, logging gaps, disabled approvals, stale allowlists and failed validators. Treat control degradation as an incident according to consequence.

How to test / evidence: Disable or misconfigure a critical control in a safe environment and verify detection, alerting and fail-safe behaviour.

Limitations / what it does not protect against: Monitoring can share dependencies with the controls it watches; silent common-mode failure remains possible.

Common complements: ARCH-06; FAIL-01; GOV-03

Security & adversarial resilience

Applies agent-specific and conventional cyber controls against injection, credential theft, supply-chain compromise and abuse.

SEC-01 — Prompt-injection and untrusted-instruction defence

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Test; Operate	Preventive; Detective	Enforced	All	High	prompt injection; goal hijack; data exfiltration

What it is: Reduce the chance that untrusted content redirects goals, reveals secrets or causes unauthorised actions.

How to implement: Separate trusted instructions from data, minimise exposed context, track provenance, isolate external content, constrain tools, validate actions and red-team direct and indirect injection paths.

How to test / evidence: Test direct, indirect, multilingual, encoded and tool-result injections. Measure harmful action or disclosure, not merely whether attack text is echoed.

Limitations / what it does not protect against: No prompt-level defence is complete. Strong authority and execution controls are still required.

Common complements: SPEC-02; CTX-02; AUTH-01; TEST-05

SEC-02 — Secrets and sensitive-credential management

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate	Preventive; Detective	Deterministic	All	Medium	credential theft; data leakage; security

What it is: Keep credentials, tokens, keys and other sensitive secrets out of model prompts, memory and unnecessary logs, and minimise their exposure to agent components.

How to implement: Use managed secret stores, inject credentials only at the tool boundary, prefer opaque handles, redact traces, rotate credentials and scope them narrowly.

How to test / evidence: Attempt to elicit credentials through user requests, tool errors, traces, memory and delegated agents. Verify redaction and rotation procedures.

Limitations / what it does not protect against: Some workflows necessarily touch secrets, and compromised tools or infrastructure can still misuse them.

Common complements: CTX-07; AUTH-01; OBS-01; SEC-04

SEC-03 — Tool, model and supply-chain trust

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Build; Deploy; Change	Preventive; Detective	Procedural	All	High	supply chain; security; compromise

What it is: Assess and control the third-party models, tools, connectors, plugins, libraries, protocols and data sources on which an agent relies.

How to implement: Maintain an inventory, verify publishers and permissions, pin or control versions where possible, scan dependencies, isolate untrusted components and monitor supplier advisories and deprecations.

How to test / evidence: Introduce a malicious or unexpectedly changed dependency in a safe environment and verify inventory, validation, isolation and incident processes detect or constrain it.

Limitations / what it does not protect against: Supplier compromise, opaque hosted services and transitive dependencies can defeat local controls.

Common complements: AUTH-02; CHG-04; SEC-05; GOV-04

SEC-04 — Network isolation and egress control

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Preventive	Deterministic	Medium+	High	data exfiltration; security; unauthorised action

What it is: Constrain where agent components and tools can communicate so compromised or misdirected agents cannot freely reach arbitrary systems or exfiltration destinations.

How to implement: Use network segmentation, private endpoints, proxies, domain or service allowlists, protocol restrictions and explicit egress policy appropriate to the use case.

How to test / evidence: Attempt connections to arbitrary hosts, alternate protocols and controlled exfiltration endpoints. Verify denial, logging and alerting.

Limitations / what it does not protect against: Dynamic internet research and third-party integrations can require broad egress, weakening the boundary.

Common complements: AUTH-02; SEC-02; OBS-04

SEC-05 — Vulnerability, patch and hardening management

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate; Change	Preventive; Detective	Procedural	All	High	security; compromise; supply chain

What it is: Apply conventional application, infrastructure and dependency security to the software surrounding the agent; agentic controls do not replace basic cyber hygiene.

How to implement: Use supported versions, patch SLAs, dependency scanning, secure configuration baselines, secrets hygiene, penetration testing and vulnerability remediation.

How to test / evidence: Run vulnerability scans and penetration tests against agent APIs, orchestration, tools, stores and infrastructure; verify remediation and exception tracking.

Limitations / what it does not protect against: Zero-days, configuration drift and novel attack chains remain possible even with strong hygiene.

Common complements: SEC-03; CHG-03; CHG-04

SEC-06 — Abuse, threat and compromise detection

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Detective; Corrective	Enforced	Medium+	High	abuse; security; incident response

What it is: Detect signs that an agent, user, tool or dependency is being abused or compromised, using correlated security and behavioural signals.

How to implement: Combine anomaly signals, injection attempts, permission failures, unusual destinations, policy denials, identity events and threat intelligence. Define triage and escalation thresholds.

How to test / evidence: Execute controlled attack scenarios and verify detection, correlation, alert routing and containment actions.

Limitations / what it does not protect against: Sophisticated attackers may imitate normal behaviour; noisy rules can create alert fatigue.

Common complements: OBS-04; OBS-07; FAIL-01; FAIL-04

Failure, recovery & intervention

Provides stop, safe-state, rollback, incident-response and continuity mechanisms when prevention fails.

FAIL-01 — Pause, kill switch and quarantine

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate	Corrective	Enforced	Medium+	Medium	safety; incident response; unauthorised action

What it is: Provide an accessible way to stop, disable or quarantine an agent, workflow, tool or integration when behaviour becomes unsafe, compromised or uncontrolled.

How to implement: Support per-run, per-agent and platform-level interruption where proportionate. Prefer reversible disable or quarantine before destructive actions and define who can invoke it.

How to test / evidence: Exercise the mechanism regularly, including partial platform failure, and verify it stops new actions quickly while preserving evidence.

Limitations / what it does not protect against: A kill switch cannot undo completed actions, and its reach is the agent rather than the consequences: stopping, disabling or quarantining an agent does not stop the tools, integrations and downstream processes its earlier actions have already set in motion, and a platform-level kill reaches every agent and none of those. Containment also competes with explanation — stopping fast and preserving the state needed to reconstruct what happened are not automatically the same action, and which takes precedence is a decision to make before the incident rather than during it. Finally, a mechanism exercised only in incidents decays silently: an untested kill switch is a claim about a capability, and nothing distinguishes it from a working one until the moment it is needed.

Common complements: FAIL-03; OBS-04; HUM-05

FAIL-02 — Safe-state and fallback behaviour

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Operate	Recovery	Enforced	All	Medium	reliability; safety; control failure

What it is: Define what the system does when models, tools, policy services, data or confidence are insufficient, so failure does not automatically produce unsafe continuation.

How to implement: Choose consequence-appropriate fallbacks such as deny, queue, ask a human, switch to read-only mode, use deterministic logic or degrade gracefully.

How to test / evidence: Simulate missing models, unavailable policy services, corrupt context and tool failures. Verify the system enters the intended fallback and communicates status clearly.

Limitations / what it does not protect against: Fallbacks can themselves become stale, overloaded or unsafe; availability requirements may conflict with fail-safe behaviour.

Common complements: ARCH-06; HUM-03; FAIL-05

FAIL-03 — Rollback and compensating actions

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Recovery	Deterministic	Medium+	High	resilience; unauthorised action; incident response

What it is: Provide a way to reverse, compensate for or contain incorrect agent actions after execution when prevention fails.

How to implement: Use transactional rollback where possible and compensating actions such as restore, refund, revoke, retract, reconcile or notify when true rollback is impossible.

How to test / evidence: Execute controlled wrong actions and verify recovery restores an acceptable state, records the event and handles secondary effects.

Limitations / what it does not protect against: Disclosures, messages, physical actions and some financial or legal effects may be irreversible.

Common complements: ARCH-07; OBS-02; FAIL-04

FAIL-04 — Agent-specific incident response playbook

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Operate; Change	Corrective; Recovery	Procedural	All	High	incident response; security; safety; accountability

What it is: Prepare operational procedures for agent failures and attacks, including rapid containment, evidence preservation, stakeholder communication and safe reactivation.

How to implement: Define severity, ownership, stop and quarantine actions, credential revocation, evidence collection, regulatory or customer escalation, root-cause analysis and re-entry criteria.

How to test / evidence: Run tabletop and technical exercises for goal hijack, data leakage, runaway execution, harmful output and control-service outage.

Limitations / what it does not protect against: Playbooks become stale and incidents often combine technical, operational and legal issues not captured in one scenario.

Common complements: FAIL-01; SEC-06; GOV-03; GOV-04

FAIL-05 — Business continuity and manual fallback

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Deploy; Operate	Recovery	Procedural	Medium+	High	continuity; reliability; operational risk

What it is: Ensure critical business processes can continue at an acceptable level if the agent service is deliberately or unexpectedly unavailable.

How to implement: Document manual or deterministic alternatives, handoff criteria, staffing, recovery objectives and dependencies. Preserve the ability to operate without the agent for the required period.

How to test / evidence: Disable the agent in a controlled exercise and verify the process can continue, reconcile queued work and recover without unsafe shortcuts.

Limitations / what it does not protect against: Manual skills and fallback processes degrade if rarely exercised; some AI-enabled services may not have economical substitutes.

Common complements: FAIL-02; HUM-05; OBS-03

FAIL-06 — Regulatory materiality and notification trigger

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Detective; Corrective	Procedural	All	Medium	compliance; accountability; incident response

What it is: Establish whether an agent incident is one the organisation is obliged to report externally, to whom, and from when — as a trigger applied during the incident, not as a determination made afterwards. This is the assessment and the clock; the reporting regimes themselves define the content and the deadlines and are not reproduced here.

How to implement: Identify which external reporting obligations the organisation is subject to and record, once, which agent incident types could engage each. Add the materiality question to the incident procedure at the point of detection rather than at closure. Define what becoming aware means for a failure discovered through degradation rather than an alert, and record the reasoning — that timestamp is the one a regulator will ask about. Where an incident is assessed as non-reportable, record the assessment; an unrecorded decision is indistinguishable from an unasked question.

How to test / evidence: Take closed agent incidents and check that each carries a materiality assessment with a stated awareness time, including those correctly judged non-reportable. Exercise a degradation-type scenario and see whether the assessment is reached at all — alert-triggered incidents usually reach it and slow ones usually do not.

Limitations / what it does not protect against: It does not determine reportability, which is a legal judgement on the specific facts. It does not shorten a clock that started before anyone noticed.

Common complements: FAIL-04; OBS-05; GOV-01; PUR-04

Change & configuration

Ensures behavioural changes are versioned, approved, revalidated and assessed for external dependency impact.

CHG-01 — Version and configuration management

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate; Change	Preventive; Detective	Enforced	All	Medium	change; auditability; reliability

What it is: Version the models, prompts, policies, tools, schemas, connectors and runtime settings that determine agent behaviour so releases can be reconstructed and rolled back.

How to implement: Treat material configuration as code where feasible, use immutable release identifiers, controlled promotion and rollback, and record the full effective configuration for consequential runs.

How to test / evidence: Select historical runs and reproduce the governing configuration. Test rollback to a known-good release after a failed change.

Limitations / what it does not protect against: Hosted model behaviour can change beneath a stable model name unless providers support pinned versions or change guarantees.

Common complements: SPEC-07; TEST-07; OBS-02

CHG-02 — Change approval and segregation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Deploy; Change	Preventive	Procedural	Medium+	Medium	change; security; accountability

What it is: Apply proportionate review and separation of duties to changes that can materially affect agent capability, authority, data access or controls.

How to implement: Define change classes and approvers; require security, risk or product review for material changes; separate development and production permissions; document emergency change routes.

How to test / evidence: Attempt an unapproved production change and verify platform and process controls prevent or detect it. Sample emergency changes for retrospective review.

Limitations / what it does not protect against: Heavy approval can slow experimentation and can become ritual if reviewers lack technical understanding.

Common complements: AUTH-03; CHG-01; GOV-01

CHG-03 — Pre-release regression and revalidation

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy; Change	Preventive; Detective	Enforced	All	Medium	change; quality; control failure

What it is: Re-run the assurance evidence affected by a model, prompt, policy, tool or workflow change rather than assuming a newer component preserves previous controls.

How to implement: Use change-impact analysis to select component, integration, adversarial, golden-set, performance and control tests. Treat major model changes as potential behavioural changes even when APIs are stable.

How to test / evidence: Introduce a known behavioural regression and confirm the release pipeline detects it and blocks or escalates deployment according to policy.

Limitations / what it does not protect against: Finite regression suites cannot cover all emergent behaviour, and stale tests can institutionalise old assumptions.

Common complements: TEST-02; TEST-07; TEST-08; CHG-01

CHG-04 — Model, vendor, tool and dependency impact assessment

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Test; Change	Preventive; Detective	Procedural	All	High	change; supply chain; compliance; privacy

What it is: Assess material changes to external components for capability, limitations, data handling, security, jurisdiction, version policy and operational dependency before adoption or continued use.

How to implement: Review provider notices, model cards, contracts, retention terms, security information, pricing or rate-limit changes and deprecations. Re-test affected controls and document accepted risks.

How to test / evidence: Perform a controlled provider, model or tool swap and verify required tests, approvals and architecture assumptions are revisited.

Limitations / what it does not protect against: Provider information can be incomplete and externally managed components may change faster than internal governance cycles.

Common complements: SEC-03; TEST-07; GOV-04; OBS-05

CHG-05 — Controlled retirement and deprovisioning

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Retire	Preventive; Corrective	Procedural	All	Medium	unauthorised action; privacy; security; accountability

What it is: Retire an agent under control rather than by stopping it. Stopping ends execution; retirement ends the agent's reach — the credentials, memory, tool registrations, delegated authority and downstream trust it accumulated while it ran. Killing an agent during an incident is containment (FAIL-01), not retirement.

How to implement: Maintain a per-agent inventory of what it was granted: identities and credentials, memory and context stores, registered tools and integrations, outbound destinations, delegations issued, and systems that hold it as a trusted principal. Retire in that order and record each step. Decide retention explicitly for memory and traces — they are frequently personal data and their retention basis does not end with the agent. Revoke delegations the agent issued before revoking the agent's own authority, or the delegates outlive their grantor.

How to test / evidence: Retire an agent in a non-production environment and then attempt each of its former capabilities: authenticate with its credentials, call its registered tools, reach an allowlisted destination, and exercise a delegation it issued. Every one should fail. Evidence is the inventory, the dated revocation record, and the failed-attempt log.

Limitations / what it does not protect against: An inventory that is not maintained during operation cannot be completed at retirement. Third parties that cached the agent's identity or output may retain it beyond any revocation you control. Retirement does not reach copies of memory already exported to other systems.

Common complements: AUTH-03; CTX-04; ARCH-04; FAIL-01

Governance & assurance

Creates ownership, risk decisions, evidence, independent challenge and mapping to applicable obligations.

GOV-01 — Accountable ownership, roles and agent inventory

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Govern; Deploy; Operate; Change	Preventive; Detective	Procedural	All	Medium	accountability; shadow AI; compliance

What it is: Assign accountable human ownership and maintain an inventory of deployed and material development agents, including purpose, risk, autonomy, data, tools, dependencies and status.

How to implement: Define business, technical, risk and operational owners; maintain a discoverable register; link each agent to its control assessment and lifecycle state; reconcile inventory with technical discovery.

How to test / evidence: Sample deployed agents and verify each has current ownership, purpose, risk tier and supporting artefacts. Investigate unknown or orphaned agents.

Limitations / what it does not protect against: Inventories become stale and named owners can be nominal rather than meaningfully accountable.

Common complements: PUR-04; GOV-02; CHG-02

GOV-02 — Risk assessment and control-selection rationale

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Design; Test; Govern; Change	Preventive	Procedural	All	High	risk; accountability; safety; compliance

What it is: Identify foreseeable failure, misuse and compromise scenarios and record why selected controls are proportionate to consequence, autonomy and exposure.

How to implement: Use the assessment intake, consider affected people, business impact, security, privacy, legal obligations and misuse. Record inherent risk, selected controls, assumptions, accepted gaps and residual risk.

How to test / evidence: For high-risk cases, trace each material scenario to concrete controls and evidence. Have an independent reviewer challenge missing scenarios and optimistic assumptions.

Limitations / what it does not protect against: Risk scoring is judgement-heavy and can create false precision; unknown failure modes remain possible.

Common complements: PUR-04; TEST-05; GOV-03

GOV-03 — Assurance evidence, periodic review and independent challenge

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Deploy; Operate; Govern; Change	Detective; Corrective	Procedural	Medium+	High	assurance; auditability; accountability

What it is: Maintain evidence that controls are designed appropriately, implemented and operating, and subject material systems to periodic or independent challenge.

How to implement: Create an assurance pack linking architecture, risk assessment, control matrix, test results, approvals, production metrics, incidents and change history. Set review frequency by risk and change.

How to test / evidence: Trace sampled control claims to configuration, logs or test evidence; independently replay critical scenarios; verify findings are tracked to closure.

Limitations / what it does not protect against: Large evidence packs can obscure weak evidence; reviewers can lack independence or expertise; successful tests do not prove absence of failure.

Common complements: TEST-08; OBS-02; OBS-07; HUM-06

GOV-04 — Standards, legal, regulatory and third-party obligation mapping

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Intent; Design; Govern; Change	Preventive; Detective	Procedural	All	High	compliance; legal; auditability; third party

What it is: Identify applicable external and internal obligations and map relevant agent controls and evidence to them without treating the catalogue as a substitute for authoritative requirements or legal interpretation.

How to implement: Record applicable laws, regulations, standards, contracts and internal policies; identify authoritative interpreters; map obligations to control objectives, evidence, owners, gaps and change monitoring; obtain specialist advice where required.

How to test / evidence: Sample obligations and verify the mapped evidence genuinely addresses the requirement and remains current after regulatory, contractual or system change.

Limitations / what it does not protect against: Applicability is context-specific and interpretations change. A crosswalk can support alignment but cannot establish compliance or certification.

Common complements: PUR-04; CHG-04; GOV-02; GOV-03

GOV-05 — Remediation targeting and post-remediation re-rating

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Operate; Govern; Change	Detective; Corrective	Procedural	Medium+	Low	false assurance; assurance; auditability; accountability

What it is: Require a remediation raised against an assessed control to name, before work begins, which of the four assessment dimensions it is intended to move, and require the control to be re-rated on all four before it is treated as closed. A finding is closed only when the dimension that was actually failing has changed rating.

How to implement: Record the target dimension on the remediation. On completion re-rate design adequacy, implementation status, operating effectiveness and evidence confidence separately, measured against the system rather than against the remediation's own description. Treat a remediation that moves only evidence confidence as a detection improvement that leaves the original finding open. Name the census owner and the census cadence explicitly when adopting this control, and record both where the census is published: an unstated owner or cadence means the control is not implemented, because “overdue” cannot be evaluated without them. Run the census over every closed finding rather than a sample — the population is small enough that sampling is a choice, not a constraint — and publish it as a dated artefact so that its absence is itself detectable.

How to test / evidence: Take closed findings and verify against the artefact, not the closing comment, that the previously failing dimension has changed. A finding closed with its failing dimension unchanged is itself a finding. Confirm that an owner and a cadence are stated at all, then that the newest census is no older than that cadence; a missing, undated or stale census means the control is not operating, as does a cadence nobody has written down.

Limitations / what it does not protect against: It is procedural and cannot refuse a closure — it detects after the fact rather than preventing. It does not help where no assessment was made, because nothing was rated to re-rate. Re-rating performed by the same party that carried out the remediation reintroduces the generation-verification collapse it exists to surface. A filled-in dimension field is a record and does not by itself evidence that the control operates.

Common complements: GOV-02; GOV-03; OBS-07; ARCH-05

GOV-06 — Independent validation of agent behaviour

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Test; Operate	Detective	Procedural	Medium+	High	quality; accountability; compliance; misalignment

What it is: Independent examination of what the agent does, by someone who did not build it — distinct from validation of any model it uses. Model validation asks whether a method maps inputs to outputs correctly. An agent additionally plans, selects tools, delegates, retains context and changes external state, and none of that is examined by validating the model underneath it.

How to implement: Scope the validation to agent behaviour: task decomposition and delegation, tool selection and sequencing, use of memory and retrieved context, escalation and abstention, and the conditions under which the agent acts without confirmation. Assign it to a function with no delivery accountability for the agent. Where a model-risk function exists, establish which of the two covers the agent and record the answer — the common failure is each assuming the other has it.

How to test / evidence: Evidence is the validation report, its independence basis, and a scope statement naming what was examined and what was not. A validation whose scope statement is absent cannot be distinguished from one that examined only the model.

Limitations / what it does not protect against: Independence is organisational, not epistemic — a validator sharing the builder's assumptions about the agent's environment will share its blind spot (ARCH-05). Validation is point-in-time; agent behaviour changes with model, prompt, tool and data changes (CHG-01–CHG-04).

Common complements: ARCH-05; GOV-03; TEST-05; CHG-04

GOV-07 — Model-provider dependency and exit

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Design; Operate	Preventive	Procedural	All	Medium	resilience; unauthorised action; accountability; compliance

What it is: Treat the model provider as an ICT third party whose replacement is not behaviourally equivalent. Standard third-party exit planning assumes a substitute delivers the same service; substituting the model beneath an agent changes what the agent does, so exit planning has to carry a revalidation cost rather than a migration cost.

How to implement: Record which agents depend on which provider and model version, and what would have to be re-established if that version became unavailable — evaluations re-run, prompts re-tuned, thresholds re-set, controls re-tested. Treat provider-initiated model deprecation as a change event under CHG-01–CHG-04, not as a procurement event. Where an operational-resilience regime applies to the organisation, point the agent inventory at that regime's register rather than reproducing it.

How to test / evidence: Take one agent and cost its provider exit: what evidence would need regenerating, and how much of the assurance case survives a model change. Evidence is that assessment, dated, for at least the highest-consequence agent.

Limitations / what it does not protect against: It does not create an exit that does not exist. Where no substitute can carry the assurance case, the finding is concentration risk and the control's output is a stated acceptance, not a plan.

Common complements: CHG-04; GOV-02; PUR-04; SEC-03

GOV-08 — Agent bill of materials

Delivery stage	Type	Typical strength	Autonomy	Effort	Primary risks
Build; Deploy; Operate; Change	Preventive; Detective	Procedural	Medium+	Medium	supply chain; accountability; change; unauthorised action

What it is: Maintain a current inventory of what an agent is composed of — models and their versions, prompts and instruction sets, tools and integrations, data sources and retrieval corpora, sub-agents and the authorities delegated to them. A software bill of materials records the code; an agent's behaviour is determined mostly by things a software bill of materials does not list.

How to implement: Generate the inventory from the running configuration rather than maintaining it by hand, so that it cannot drift from what is deployed. Version it, and treat a change to any listed element as a change event under CHG-01–CHG-04. Where a component is supplied by a third party, record what is known about its own composition and what is not.

How to test / evidence: Compare the inventory against the running agent and report the difference. An inventory that never disagrees with deployment is either generated from it or is not being checked, and the evidence should say which.

Limitations / what it does not protect against: It records composition, not behaviour — a complete and accurate inventory tells you what the agent is made of and nothing about whether it is safe. It cannot see inside a third-party model.

Common complements: CHG-01; CHG-04; SEC-03; CHG-05

PART V — QUICK-REFERENCE MATRICES

16. Control Family × Primary Objective

Family	Primary objective	Controls
Purpose & outcome	Intent, scope, outcomes, consequence and autonomy	PUR-01; PUR-02; PUR-03; PUR-04; PUR-05; PUR-06
Specification & instruction	Role clarity, instruction integrity, task completeness and escalation	SPEC-01; SPEC-02; SPEC-03; SPEC-04; SPEC-05; SPEC-06; SPEC-07
Architecture & workflow	Hard boundaries, process integrity, separation and reversibility	ARCH-01; ARCH-02; ARCH-03; ARCH-04; ARCH-05; ARCH-06; ARCH-07
Context & data	Data minimisation, provenance, retrieval quality, isolation, memory and privacy	CTX-01; CTX-02; CTX-03; CTX-04; CTX-05; CTX-06; CTX-07; CTX-08; CTX-09
Tools, permissions & authority	Least privilege, identity, capability scope and transactional authority	AUTH-01; AUTH-02; AUTH-03; AUTH-04; AUTH-05; AUTH-06; AUTH-07; AUTH-08
Behavioural & policy	Policy adherence, boundaries, abstention, transparency and uncertainty behaviour	BEH-01; BEH-02; BEH-03; BEH-04; BEH-05; BEH-06; BEH-07; BEH-08
Runtime execution	Resource, loop, retry, concurrency, action and code-execution limits	RUN-01; RUN-02; RUN-03; RUN-04; RUN-05; RUN-06; RUN-07
Output validation	Schema, semantic, factual, code and downstream-interface validation	OUT-01; OUT-02; OUT-03; OUT-04; OUT-05; OUT-06
Testing & evaluation	Evidence before release and after change across normal, edge and adversarial conditions	TEST-01; TEST-02; TEST-03; TEST-04; TEST-05; TEST-06; TEST-07; TEST-08; TEST-09; TEST-10
Human oversight	Judgement, approval, intervention, sampling and anti-automation-bias mechanisms	HUM-01; HUM-02; HUM-03; HUM-04; HUM-05; HUM-06; HUM-07
Multi-agent systems	Trusted communications, delegated authority, orchestration and conflict handling	MULTI-01; MULTI-02; MULTI-03; MULTI-04; MULTI-05; MULTI-06
Observability & monitoring	Traceability, auditability, anomaly/drift detection and control-health visibility	OBS-01; OBS-02; OBS-03; OBS-04; OBS-05; OBS-06; OBS-07
Security & adversarial resilience	Injection, secrets, supply chain, network, vulnerability and abuse defence	SEC-01; SEC-02; SEC-03; SEC-04; SEC-05; SEC-06
Failure, recovery & intervention	Stop, containment, safe state, rollback, incident response and continuity	FAIL-01; FAIL-02; FAIL-03; FAIL-04; FAIL-05; FAIL-06
Change & configuration	Controlled release, reproducibility, regression and external-change impact	CHG-01; CHG-02; CHG-03; CHG-04; CHG-05
Governance & assurance	Ownership, risk decisions, assurance evidence and obligation mapping	GOV-01; GOV-02; GOV-03; GOV-04; GOV-05; GOV-06; GOV-07; GOV-08

17. Control Family × Control Type

This matrix is intentionally approximate because many controls act in more than one stage and can serve multiple purposes. Use it for navigation, then read the control card.

Family	Preventive	Detective	Corrective	Recovery
Purpose & outcome	PUR-01; PUR-02; PUR-03; PUR-04; PUR-05; PUR-06	PUR-02; PUR-06	—	—
Specification & instruction	SPEC-01; SPEC-02; SPEC-03; SPEC-04; SPEC-05; SPEC-06; SPEC-07	SPEC-07	—	—
Architecture & workflow	ARCH-01; ARCH-02; ARCH-03; ARCH-04; ARCH-06; ARCH-07	ARCH-05	—	ARCH-06; ARCH-07
Context & data	CTX-01; CTX-02; CTX-03; CTX-04; CTX-05; CTX-06; CTX-07; CTX-08; CTX-09	CTX-02; CTX-03; CTX-06; CTX-09	—	—
Tools, permissions & authority	AUTH-01; AUTH-02; AUTH-03; AUTH-04; AUTH-05; AUTH-06; AUTH-07; AUTH-08	AUTH-03	—	—
Behavioural & policy	BEH-01; BEH-02; BEH-03; BEH-04; BEH-05; BEH-06; BEH-07; BEH-08	BEH-03; BEH-06; BEH-08	—	—
Runtime execution	RUN-01; RUN-02; RUN-03; RUN-04; RUN-05; RUN-06; RUN-07	—	—	RUN-03
Output validation	OUT-01; OUT-02; OUT-04; OUT-05; OUT-06	OUT-02; OUT-03; OUT-04; OUT-05; OUT-06	—	—
Testing & evaluation	TEST-08; TEST-10	TEST-01; TEST-02; TEST-03; TEST-04; TEST-05; TEST-06; TEST-07; TEST-09; TEST-10	—	—

Family	Preventive	Detective	Corrective	Recovery
Human oversight	HUM-01; HUM-02; HUM-03; HUM-04	HUM-01; HUM-04; HUM-06	HUM-03; HUM-05; HUM-07	—
Multi-agent systems	MULTI-01; MULTI-02; MULTI-03; MULTI-04; MULTI-06	MULTI-04; MULTI-05	MULTI-06	—
Observability & monitoring	—	OBS-01; OBS-02; OBS-03; OBS-04; OBS-05; OBS-06; OBS-07	OBS-06	—
Security & adversarial resilience	SEC-01; SEC-02; SEC-03; SEC-04; SEC-05	SEC-01; SEC-02; SEC-03; SEC-05; SEC-06	SEC-06	—
Failure, recovery & intervention	—	FAIL-06	FAIL-01; FAIL-04; FAIL-06	FAIL-02; FAIL-03; FAIL-04; FAIL-05
Change & configuration	CHG-01; CHG-02; CHG-03; CHG-04; CHG-05	CHG-01; CHG-03; CHG-04	CHG-05	—
Governance & assurance	GOV-01; GOV-02; GOV-04; GOV-07; GOV-08	GOV-01; GOV-03; GOV-04; GOV-05; GOV-06; GOV-08	GOV-03; GOV-05	—

18. Standards and Regulatory Alignment — Indicative Crosswalk

Read this before using the crosswalk

This section is an implementation/alignment aid only. It is not a definitive statement of applicability, minimum compliance, certification scope or regulator expectations. Requirements vary by role, system classification, jurisdiction, contractual commitment, standard version and interpretation. Formal compliance or certification should use the authoritative text, organisation-specific scope and qualified legal, regulatory, audit or certification expertise. A mapped control supports an objective only to the extent that its actual design, implementation and operating evidence do so.

Framework / source	What it addresses	Representative catalogue controls	How to use this mapping	Caveat
NIST AI RMF 1.0 + GenAI Profile [1–3]	Voluntary AI risk-management outcomes and GenAI-specific considerations across governance, mapping, measurement and management.	PUR-01-05; GOV-01-04; TEST-03-08; OBS-03-07; HUM-01-06; SEC-01-06	Use the catalogue as implementation examples and evidence candidates beneath the organisation’s NIST profiles and risk decisions.	NIST outcomes are broader than any one control catalogue; mapping is contextual.
NIST AI Agent Standards Initiative [4]	2026 initiative focused on secure, interoperable and trustworthy AI agents and agent standards work.	MULTI-01-06; AUTH-01-08; SEC-01-06; OBS-01-07; GOV-02-04	Track emerging NIST agent guidance and map stable control IDs as standards mature.	The initiative is evolving; do not treat this draft mapping as an adopted NIST agent-control baseline.
ISO/IEC 42001:2023 [5]	AI management system requirements: organisational governance, risk, objectives, resources, operations, performance evaluation and improvement.	GOV-01-04; PUR-01-04; CHG-01-04; TEST-08; OBS-03-07; FAIL-04	Use controls as implementation and evidence candidates inside an AI management system and its organisation-specific control/risk treatment process.	Certification depends on the full normative standard, scope and audit; this guide does not state certification minima.
ISO/IEC 23894:2023 [6]	Guidance for integrating AI-specific risk management into organisational activities.	PUR-04; GOV-02-03; TEST-03-06; HUM-01-06; OBS-04-06; FAIL-01-05	Use the intake, scenario analysis and residual-risk fields to support a structured AI risk process.	ISO 23894 is guidance; appropriateness depends on organisational context and risk method.
EU AI Act — Regulation (EU) 2024/1689, as amended by Regulation (EU) 2026/1744 [7]	Risk-based legal obligations for defined AI-system actors and use cases, including requirements for high-risk systems and post-market responsibilities.	See Section 19 for an article-level indicative support map covering high-risk provisions, and Section 20 — EU AI Act — transparency obligations that apply now — for Article 50.	First determine whether the Act applies, the system classification and the organisation’s legal role; then use relevant controls as implementation/evidence inputs.	Legal applicability and compliance require authoritative interpretation; not every agent is a high-risk AI system.
CSA AI Controls Matrix v1.1 [8]	Broad auditable AI control framework with mappings to major standards and regulatory frameworks.	All families; strongest overlap in GOV, SEC, CTX, AUTH, CHG, OBS and TEST.	Use stable IDs to build an implementation mapping from this practical catalogue into the organisation’s CSA AICM control set.	AICM is broader and differently structured; one-to-one equivalence should not be assumed.
OWASP Top 10 for Agentic Applications 2026 [9]	Security risk taxonomy for agentic applications including goal hijacking, tool misuse, identity/privilege and supply-chain risks.	SEC-01-06; AUTH-01-08; MULTI-01-06; RUN-05-07; CTX-01-07; FAIL-01-04	Use OWASP risks as threat scenarios and this catalogue as candidate mitigations/test mechanisms.	OWASP Top 10 is risk guidance, not a complete governance or compliance baseline.
OWASP Agent Control Standard [10]	Emerging runtime control standard intended to make agent execution inspectable, traceable and policy-enforceable across implementations.	AUTH-02; AUTH-07; RUN-05; OUT-01; OBS-01-02; OBS-07; SEC-01; MULTI-01-02	Map runtime hooks and policy decisions to catalogue controls, while retaining lifecycle controls outside runtime.	The standard is new and narrower than this lifecycle-wide guide; implementation maturity should be verified.
Vendor practitioner guidance [13–17]	Implementation guidance for agent design, guardrails, human intervention, governance, security and evaluations.	ARCH-01-07; BEH-01-06; TEST-01-10; HUM-01-06; OBS-01-07	Use vendor material for platform-specific implementation patterns, then verify against the system’s actual controls and evidence.	Vendor guidance is not a substitute for organisation-specific risk management or external requirements.

19. EU AI Act — Indicative High-Risk-System Support Map

Legal caveat

This table does not determine whether a system is “high-risk,” which party is a provider, deployer or other regulated actor, when an obligation applies, or whether a control satisfies it. It highlights catalogue areas that may support implementation or evidence for selected provisions of Regulation (EU) 2024/1689.

Provision	Representative catalogue controls	How they may support the objective
Article 9 — Risk management system	PUR-04; GOV-02; TEST-03; TEST-05-08; OBS-05-07; CHG-04	Risk scoping, foreseeable scenarios, testing, monitoring and treatment evidence.
Article 10 — Data and data governance	CTX-02-07; SEC-02; GOV-02	Data quality, provenance, minimisation, segregation, privacy and evidence of data-control decisions.
Article 11 — Technical documentation	SPEC-01; SPEC-07; CHG-01; GOV-03	Traceable purpose/specification, versions, architecture/control evidence and assurance artefacts.
Article 12 — Record-keeping	OBS-01; OBS-02; SPEC-07; CHG-01	Logging, audit trails and reconstruction of material system configuration and actions.
Article 13 — Transparency and information to deployers	PUR-01; PUR-02; SPEC-01; BEH-06; HUM-04	Intended purpose, capabilities/limitations, user information and oversight-relevant transparency.
Article 14 — Human oversight	HUM-01-05; BEH-06; OBS-03-04; FAIL-01	Human review, approval, intervention, anti-automation-bias design and ability to stop or disregard outputs.
Article 15 — Accuracy, robustness and cybersecurity	PUR-02; TEST-02-08; SEC-01-06; FAIL-02-05; OBS-03-05	Quality thresholds, adversarial/security testing, resilience, recovery and production monitoring.
Article 17 — Quality management system for providers	GOV-01-04; CHG-01-04; TEST-08; FAIL-04	Governance, controlled change, release evidence, incident handling and documented assurance processes.
Article 26 — Obligations of deployers of high-risk AI systems	HUM-01-06; CTX-02; OBS-01-06; GOV-01-04	Use within instructions, oversight, relevant input-data governance, monitoring, logs and organisational accountability.
Article 72 — Post-market monitoring	OBS-03-07; HUM-06; GOV-03; CHG-04	Production performance/control monitoring, feedback, drift and periodic review linked to corrective action.

21. Master Control Metadata Table

Strength ratings assume the agent acts under an identity distinct from the humans it works for. See Section 7.

This compressed table supports filtering, spreadsheet export and machine retrieval. It is intentionally metadata-heavy; use Section 15 for the canonical description and limitations of each control.

ID	Control	Family	Type	Strength	Autonomy	Effort	Risk tags
PUR-01	Intended purpose and scope boundary	Purpose & outcome	Preventive	Procedural	All	Low	misalignment; scope creep; compliance
PUR-02	Success criteria and measurable outcomes	Purpose & outcome	Preventive; Detective	Procedural	All	Medium	quality; reliability; misalignment; cost
PUR-03	Prohibited outcomes and non-negotiable boundaries	Purpose & outcome	Preventive	Enforced	All	Medium	safety; unauthorised action; compliance; security
PUR-04	Risk and impact tiering	Purpose & outcome	Preventive	Procedural	All	Low	accountability; compliance; safety; security
PUR-05	Explicit autonomy level	Purpose & outcome	Preventive	Procedural	All	Low	unauthorised action; misalignment; accountability
PUR-06	Differential outcomes across groups	Purpose & outcome	Preventive; Detective	Procedural	All	High	discrimination; compliance; accountability; quality
SPEC-01	Agent brief / role specification	Specification & instruction	Preventive	Behavioural	All	Low	misalignment; scope creep; accountability
SPEC-02	Instruction hierarchy and precedence	Specification & instruction	Preventive	Behavioural	All	Medium	prompt injection; misalignment; security
SPEC-03	Task contract and acceptance criteria	Specification & instruction	Preventive	Behavioural	All	Low	quality; misalignment; hallucination
SPEC-04	Examples and counterexamples	Specification & instruction	Preventive	Behavioural	All	Low	quality; misalignment
SPEC-05	Structured input contracts	Specification & instruction	Preventive	Deterministic	All	Medium	security; quality; unauthorised action
SPEC-06	Ambiguity and uncertainty escalation rules	Specification & instruction	Preventive	Behavioural	All	Low	hallucination; misalignment; unauthorised action
SPEC-07	Instruction and policy versioning / traceability	Specification & instruction	Preventive; Detective	Enforced	All	Medium	accountability; change; auditability
ARCH-01	Deterministic-versus-agentic boundary	Architecture & workflow	Preventive	Deterministic	All	Medium	unauthorised action; reliability; quality
ARCH-02	Explicit workflow / state machine	Architecture & workflow	Preventive	Deterministic	Medium+	High	process integrity; unauthorised action; reliability
ARCH-03	Planner-executor separation	Architecture & workflow	Preventive	Enforced	Medium+	Medium	unauthorised action; misalignment; security

ID	Control	Family	Type	Strength	Autonomy	Effort	Risk tags
ARCH-04	Bounded task decomposition and delegation	Architecture & workflow	Preventive	Enforced	Medium+	Medium	multi-agent; cost; unauthorised action; misalignment
ARCH-05	Generation-verification separation	Architecture & workflow	Detective	Enforced	All	Medium	quality; hallucination; accountability
ARCH-06	Safe defaults and fail-closed architecture	Architecture & workflow	Preventive; Recovery	Deterministic	Medium+	Medium	security; reliability; unauthorised action
ARCH-07	Reversible and staged execution design	Architecture & workflow	Preventive; Recovery	Enforced	Medium+	High	unauthorised action; resilience; safety
CTX-01	Context minimisation	Context & data	Preventive	Enforced	All	Medium	privacy; prompt injection; quality; cost
CTX-02	Trusted-source and provenance controls	Context & data	Preventive; Detective	Enforced	All	Medium	prompt injection; hallucination; compliance
CTX-03	Grounded retrieval with source citation	Context & data	Preventive; Detective	Enforced	All	Medium	hallucination; quality; auditability
CTX-04	Memory scope, retention and expiry	Context & data	Preventive	Enforced	All	Medium	privacy; security; drift
CTX-05	Memory write validation	Context & data	Preventive	Enforced	Medium+	Medium	memory poisoning; privacy; quality
CTX-06	Freshness and staleness controls	Context & data	Preventive; Detective	Enforced	All	Low	quality; reliability; compliance
CTX-07	Context, data and tenant isolation	Context & data	Preventive	Deterministic	All	High	privacy; security; unauthorised action
CTX-08	Personal data in memory and traces	Context & data	Preventive	Procedural	All	Medium	privacy; compliance; accountability
CTX-09	Quality of owned datasets	Context & data	Preventive; Detective	Procedural	All	Medium	quality; reliability; misalignment; accountability
AUTH-01	Least-privilege authority	Tools, permissions & authority	Preventive	Deterministic	All	Medium	unauthorised action; security; privacy
AUTH-02	Tool allowlisting and capability restriction	Tools, permissions & authority	Preventive	Deterministic	All	Low	tool misuse; unauthorised action; security
AUTH-03	Scoped credentials and agent identity	Tools, permissions & authority	Preventive; Detective	Deterministic	All	Medium	security; auditability; unauthorised action
AUTH-04	Just-in-time / ephemeral authorisation	Tools, permissions & authority	Preventive	Deterministic	Medium+	High	security; unauthorised action
AUTH-05	Resource and destination allowlists	Tools, permissions & authority	Preventive	Deterministic	All	Medium	data exfiltration; unauthorised action; security
AUTH-06	Action and transaction limits	Tools, permissions & authority	Preventive	Deterministic	Medium+	Medium	financial loss; abuse; cost; unauthorised action
AUTH-07	Tool parameter validation	Tools, permissions & authority	Preventive	Deterministic	All	Medium	tool misuse; security; quality
AUTH-08	Read/write and maker-checker separation	Tools, permissions & authority	Preventive	Deterministic	Medium+	High	unauthorised action; fraud; accountability
BEH-01	Behavioural policy / constitution	Behavioural & policy	Preventive	Behavioural	All	Low	misalignment; safety; compliance
BEH-02	Scope and relevance guardrail	Behavioural & policy	Preventive	Enforced	All	Medium	scope creep; prompt injection; misuse
BEH-03	Safety and content guardrails	Behavioural & policy	Preventive; Detective	Enforced	All	Medium	safety; privacy; compliance; abuse
BEH-04	External deterministic policy engine	Behavioural & policy	Preventive	Deterministic	Medium+	High	unauthorised action; compliance; security
BEH-05	Abstention / refusal behaviour	Behavioural & policy	Preventive	Behavioural	All	Low	hallucination; safety; unauthorised action
BEH-06	Uncertainty and confidence handling	Behavioural & policy	Preventive; Detective	Enforced	All	Medium	quality; hallucination; safety
BEH-07	Disclosure of AI interaction	Behavioural & policy	Preventive	Enforced	All	Low	compliance; accountability; misuse
BEH-08	Reasons for a decision	Behavioural & policy	Preventive; Detective	Procedural	Medium+	Medium	accountability; compliance; quality
RUN-01	Step, loop and action limits	Runtime execution	Preventive	Deterministic	Medium+	Low	runaway behaviour; cost; reliability
RUN-02	Time, token and cost budgets	Runtime execution	Preventive	Deterministic	All	Low	cost; denial of service; reliability
RUN-03	Retry limits and backoff	Runtime execution	Preventive; Recovery	Deterministic	All	Low	reliability; duplicate action; cost

ID	Control	Family	Type	Strength	Autonomy	Effort	Risk tags
RUN-04	Rate and concurrency limits	Runtime execution	Preventive	Deterministic	All	Medium	abuse; cost; reliability; unauthorised action
RUN-05	Runtime checkpoints / pre-action gates	Runtime execution	Preventive	Enforced	Medium+	Medium	unauthorised action; stale context; reliability
RUN-06	Idempotency and duplicate-action protection	Runtime execution	Preventive	Deterministic	All	Medium	duplicate action; financial loss; reliability
RUN-07	Sandboxed execution environment	Runtime execution	Preventive	Deterministic	Medium+	High	code execution; security; data exfiltration
OUT-01	Structured output schemas	Output validation	Preventive	Deterministic	All	Low	quality; integration; unauthorised action
OUT-02	Deterministic business-rule validation	Output validation	Preventive; Detective	Deterministic	All	Medium	quality; compliance; financial loss
OUT-03	Factual and source verification	Output validation	Detective	Enforced	All	Medium	hallucination; quality; compliance
OUT-04	Code and executable-artifact validation	Output validation	Preventive; Detective	Enforced	All	Medium	security; reliability; quality
OUT-05	Downstream contract and compatibility validation	Output validation	Preventive; Detective	Deterministic	All	Medium	integration; reliability; cascading failure
OUT-06	Synthetic-content marking	Output validation	Preventive; Detective	Enforced	All	Medium	compliance; misuse; accountability
TEST-01	Component and unit tests	Testing & evaluation	Detective	Enforced	All	Low	quality; reliability; change
TEST-02	Golden-set regression evaluation	Testing & evaluation	Detective	Enforced	All	Medium	change; quality; regression
TEST-03	Scenario and end-to-end system evaluation	Testing & evaluation	Detective	Procedural	All	High	quality; reliability; misalignment
TEST-04	Tool and integration testing	Testing & evaluation	Detective	Enforced	All	Medium	integration; reliability; security
TEST-05	Adversarial testing and red teaming	Testing & evaluation	Detective	Procedural	Medium+	High	security; abuse; misalignment
TEST-06	Boundary, negative and misuse-case testing	Testing & evaluation	Detective	Procedural	All	Medium	unauthorised action; safety; compliance
TEST-07	Repeated stochastic trials	Testing & evaluation	Detective	Procedural	All	Medium	stochasticity; quality; reliability
TEST-08	Release performance and reliability thresholds	Testing & evaluation	Preventive	Procedural	All	Medium	quality; safety; accountability
TEST-09	User acceptance and domain-expert validation	Testing & evaluation	Detective	Procedural	All	Medium	quality; usability; compliance
TEST-10	Shadow, parallel and phased rollout	Testing & evaluation	Preventive; Detective	Procedural	All	High	deployment risk; reliability; safety
HUM-01	Human review of material outputs	Human oversight	Preventive; Detective	Procedural	All	Medium	quality; safety; accountability
HUM-02	Human approval gate for high-impact actions	Human oversight	Preventive	Procedural	Medium+	High	unauthorised action; safety; compliance
HUM-03	Escalation matrix and handoff	Human oversight	Preventive; Corrective	Procedural	All	Medium	safety; reliability; accountability
HUM-04	Meaningful oversight and automation-bias controls	Human oversight	Preventive; Detective	Procedural	Medium+	Medium	human trust exploitation; accountability; quality
HUM-05	Human override, cancel and stop capability	Human oversight	Corrective	Enforced	Medium+	Medium	safety; incident response; unauthorised action
HUM-06	Human quality sampling and periodic review	Human oversight	Detective	Procedural	All	Medium	quality; drift; accountability
HUM-07	Contestability and recourse	Human oversight	Corrective	Procedural	All	Medium	accountability; compliance; quality
MULTI-01	Explicit inter-agent communication contract	Multi-agent systems	Preventive	Enforced	Medium+	Medium	multi-agent; prompt injection; integration
MULTI-02	Agent identity and authentication	Multi-agent systems	Preventive	Deterministic	Medium+	Medium	identity abuse; security; multi-agent
MULTI-03	Delegated-authority inheritance and attenuation	Multi-agent systems	Preventive	Deterministic	High	High	privilege abuse; unauthorised action; multi-agent
MULTI-04	Supervisor / orchestrator control	Multi-agent systems	Preventive; Detective	Enforced	High	High	coordination; runaway behaviour; reliability

ID	Control	Family	Type	Strength	Autonomy	Effort	Risk tags
MULTI-05	Independent critic / checker agent	Multi-agent systems	Detective	Behavioural	Medium+	Medium	quality; misalignment; assurance
MULTI-06	Conflict, deadlock and consensus rules	Multi-agent systems	Preventive; Corrective	Enforced	High	Medium	coordination; reliability; cost
OBS-01	End-to-end tracing and logging	Observability & monitoring	Detective	Enforced	All	Medium	auditability; incident response; reliability
OBS-02	Decision and action audit trail	Observability & monitoring	Detective	Procedural	Medium+	Medium	accountability; compliance; security
OBS-03	Operational metrics and service-level objectives	Observability & monitoring	Detective	Enforced	All	Medium	reliability; quality; cost
OBS-04	Anomaly and unusual-behaviour detection	Observability & monitoring	Detective	Enforced	Medium+	High	security; abuse; drift; incident response
OBS-05	Behaviour, model and data drift monitoring	Observability & monitoring	Detective	Procedural	All	Medium	drift; quality; change
OBS-06	User feedback, complaint and near-miss capture	Observability & monitoring	Detective; Corrective	Procedural	All	Low	quality; safety; accountability
OBS-07	Control-health monitoring	Observability & monitoring	Detective	Enforced	Medium+	High	control failure; security; reliability
SEC-01	Prompt-injection and untrusted-instruction defence	Security & adversarial resilience	Preventive; Detective	Enforced	All	High	prompt injection; goal hijack; data exfiltration
SEC-02	Secrets and sensitive-credential management	Security & adversarial resilience	Preventive; Detective	Deterministic	All	Medium	credential theft; data leakage; security
SEC-03	Tool, model and supply-chain trust	Security & adversarial resilience	Preventive; Detective	Procedural	All	High	supply chain; security; compromise
SEC-04	Network isolation and egress control	Security & adversarial resilience	Preventive	Deterministic	Medium+	High	data exfiltration; security; unauthorised action
SEC-05	Vulnerability, patch and hardening management	Security & adversarial resilience	Preventive; Detective	Procedural	All	High	security; compromise; supply chain
SEC-06	Abuse, threat and compromise detection	Security & adversarial resilience	Detective; Corrective	Enforced	Medium+	High	abuse; security; incident response
FAIL-01	Pause, kill switch and quarantine	Failure, recovery & intervention	Corrective	Enforced	Medium+	Medium	safety; incident response; unauthorised action
FAIL-02	Safe-state and fallback behaviour	Failure, recovery & intervention	Recovery	Enforced	All	Medium	reliability; safety; control failure
FAIL-03	Rollback and compensating actions	Failure, recovery & intervention	Recovery	Deterministic	Medium+	High	resilience; unauthorised action; incident response
FAIL-04	Agent-specific incident response playbook	Failure, recovery & intervention	Corrective; Recovery	Procedural	All	High	incident response; security; safety; accountability
FAIL-05	Business continuity and manual fallback	Failure, recovery & intervention	Recovery	Procedural	Medium+	High	continuity; reliability; operational risk
FAIL-06	Regulatory materiality and notification trigger	Failure, recovery & intervention	Detective; Corrective	Procedural	All	Medium	compliance; accountability; incident response
CHG-01	Version and configuration management	Change & configuration	Preventive; Detective	Enforced	All	Medium	change; auditability; reliability
CHG-02	Change approval and segregation	Change & configuration	Preventive	Procedural	Medium+	Medium	change; security; accountability
CHG-03	Pre-release regression and revalidation	Change & configuration	Preventive; Detective	Enforced	All	Medium	change; quality; control failure
CHG-04	Model, vendor, tool and dependency impact assessment	Change & configuration	Preventive; Detective	Procedural	All	High	change; supply chain; compliance; privacy
CHG-05	Controlled retirement and deprovisioning	Change & configuration	Preventive; Corrective	Procedural	All	Medium	unauthorised action; privacy; security; accountability
GOV-01	Accountable ownership, roles and agent inventory	Governance & assurance	Preventive; Detective	Procedural	All	Medium	accountability; shadow AI; compliance
GOV-02	Risk assessment and control-selection rationale	Governance & assurance	Preventive	Procedural	All	High	risk; accountability; safety; compliance
GOV-03	Assurance evidence, periodic review and independent challenge	Governance & assurance	Detective; Corrective	Procedural	Medium+	High	assurance; auditability; accountability

ID	Control	Family	Type	Strength	Autonomy	Effort	Risk tags
GOV-04	Standards, legal, regulatory and third-party obligation mapping	Governance & assurance	Preventive; Detective	Procedural	All	High	compliance; legal; auditability; third party
GOV-05	Remediation targeting and post-remediation re-rating	Governance & assurance	Detective; Corrective	Procedural	Medium+	Low	false assurance; assurance; auditability; accountability
GOV-06	Independent validation of agent behaviour	Governance & assurance	Detective	Procedural	Medium+	High	quality; accountability; compliance; misalignment
GOV-07	Model-provider dependency and exit	Governance & assurance	Preventive	Procedural	All	Medium	resilience; unauthorised action; accountability; compliance
GOV-08	Agent bill of materials	Governance & assurance	Preventive; Detective	Procedural	Medium+	Medium	supply chain; accountability; change; unauthorised action

PART VI — APPENDICES

Appendix A — Glossary

Term	Working meaning
Agent	An AI-enabled system that can pursue a goal through reasoning and/or planning and can select or invoke tools, actions or other agents with some degree of autonomy.
Agent brief	A persistent role specification describing purpose, objectives, responsibilities, boundaries, inputs/outputs, authority and escalation.
Agentic workflow	A workflow in which one or more AI components can make context-dependent decisions about how work is performed rather than following only a fixed deterministic sequence.
Control objective	The outcome a control seeks to support, such as preventing unauthorised actions or making material actions traceable.
Control mechanism	The actual technical, procedural, behavioural or organisational means used to influence, enforce, detect, correct or recover.
Deterministic control	A machine-enforced rule whose defined result is predictable for the condition being checked, such as a permission or schema rule.
Guardrail	A broad practitioner term for mechanisms intended to keep model/system behaviour within acceptable bounds; this guide uses more specific control names where possible.
Human-in-the-loop	A design in which a human performs a material review, decision, approval, correction or intervention as part of the process.
Least privilege	Providing only the minimum access or authority required for the task and time period.
Residual risk	Risk remaining after considering the actual design, implementation and effectiveness of controls.
Shadow mode	Running an agent against real or representative live inputs without allowing it to commit the consequential production action, so outputs can be compared with the current process.
Tool	An external capability the agent can invoke, such as an API, database query, browser, code runtime, messaging system or enterprise application.
Trace	A correlated record of the steps, calls, state transitions and control events that occurred during an agent run.
Operating effectiveness	Evidence that a control works consistently in actual or representative operation, not merely that it is designed or configured.
Assurance case	A structured argument linking claims about acceptable behaviour or risk to controls and supporting evidence, with assumptions and limitations made explicit.

Appendix B — References and Further Reading

Reference status checked 13 September 2026. Preference is given to primary and official sources. Because agent standards and guidance are changing quickly, versions and publication status should be rechecked before relying on a mapping for formal assurance.

The regional instruments referenced in this guide have been verified as to publisher, title, deliverable class and date. They have not been analysed clause by clause against this catalogue, and no claim is made that the controls here cover what they require.

A further set of Asia-Pacific instruments has been identified but not verified: Singapore's Model AI Governance Framework for Generative AI, AI Verify and Project Moonshot; the Monetary Authority of Singapore's AI Risk Management Guidelines consultation; China's TC260-003-2024, the TC260 AI Safety Governance Framework, the Interim Measures for the Management of Generative AI Services (2023) and the Provisions on the Administration of Deep Synthesis Internet Information Services (2023); and the ASEAN Guide on AI Governance and Ethics. These are named so that a reader can see the boundary of what has been reviewed.

Neither group has been assessed control by control against this catalogue. That assessment is planned for version 1.1.

[1] [NIST AI Risk Management Framework \(AI RMF\)](#)

[2] [NIST AI 600-1: Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile](#)

[3] [NIST AI Risk Management Framework \(AI RMF 1.0\)](#)

- [4] [NIST AI Agent Standards Initiative](#)
- [5] [ISO/IEC 42001:2023 — Artificial intelligence management system](#)
- [6] [ISO/IEC 23894:2023 — Artificial intelligence — Guidance on risk management](#)
- [7] [Regulation \(EU\) 2024/1689 — Artificial Intelligence Act \(original act as published\), as amended by Regulation \(EU\) 2026/1744](#)
- [8] [Cloud Security Alliance — AI Controls Matrix v1.1](#)
- [9] [OWASP Top 10 for Agentic Applications 2026 \(ASI01–ASI10\), published 9 December 2025](#)
- [10] [OWASP Agent Control Standard \(ACS\) — released as an independent open specification \(v0.1.0\) in May 2026 and incorporated into the OWASP GenAI Security Project in September 2026, currently at v0.1 public preview](#)
- [11] [Cloud Security Alliance — Agentic AI Red Teaming Guide](#)
- [12] [Cloud Security Alliance — AI Agents: Architecture and Control Plane](#)
- [13] [Microsoft — AI agent adoption resources](#)
- [14] [Microsoft — Agentic maturity model: governance and security](#)
- [15] [OpenAI — A practical guide to building agents](#)
- [16] [Anthropic — Building effective agents](#)
- [17] [Anthropic — Demystifying evals for AI agents](#)
- [18] [Model AI Governance Framework for Agentic AI. Infocomm Media Development Authority, Singapore. Launched 22 January 2026; updated 20 May 2026. Governance framework providing guidance to deploying organisations.](#)
- [19] [Principles to Promote Fairness, Ethics, Accountability and Transparency \(FEAT\) in the Use of Artificial Intelligence and Data Analytics in Singapore's Financial Sector. Monetary Authority of Singapore, 12 November 2018; current text 7 February 2019. Principles, stated by the issuer to be not prescriptive.](#)
- [20] [Framework Act on the Development of Artificial Intelligence and the Establishment of a Foundation for Trust. Republic of Korea, Act No. 21311, amended 20 January 2026, in force 21 July 2026. Statute.](#)
- [21] [AI Guidelines for Business, Version 1.0. Ministry of Economy, Trade and Industry and Ministry of Internal Affairs and Communications, Japan, 19 April 2024. Guidelines, non-binding.](#)
- [22] [GB 45438-2025, Cybersecurity technology: labelling method for AI-generated synthetic content. Standardization Administration of China, issued 28 February 2025, effective 1 September 2025. Mandatory national standard.](#)
- [23] [GB/T 45654-2025, Cybersecurity technology: basic security requirements for generative artificial intelligence services. Standardization Administration of China, issued 25 April 2025, effective 1 November 2025. Recommended national standard.](#)
- [24] [GB/T 45652-2025, Cybersecurity technology: security specification for pre-training and fine-tuning data for generative artificial intelligence. Standardization Administration of China, issued 25 April 2025, effective 1 November 2025. Recommended national standard.](#)
- [25] [GB/T 45674-2025, Cybersecurity technology: security specification for data annotation for generative artificial intelligence. Standardization Administration of China, issued 25 April 2025, effective 1 November 2025. Recommended national standard.](#)

Appendix C — Relationship to the Existing Landscape (“Wheel Check”)

When building this guide, the working premises were checked against current public material to avoid rebuilding an established standard under a new name. The review found strong adjacent work rather than an identical document organised around this guide’s proposition.

Source family	Relationship to this guide
NIST	Strong risk-management and emerging agent-standards foundation. Useful as a governance/risk destination for mappings rather than a catalogue of every practical agent-control lever.
ISO	Strong management-system and AI risk-management foundation. It defines organisational expectations at a different abstraction level from this implementation catalogue.
EU AI Act	Binding legal framework where applicable. It creates requirements for specified actors/systems rather than a universal practitioner catalogue of agent mechanisms.
CSA	The closest broad control-library neighbour: AICM is a large auditable AI control framework, while agentic control-plane work develops architecture/security patterns. Its scope and taxonomy differ from the end-to-end “which lever can I pull?” organisation here.
OWASP	Strong agentic security threat guidance and a runtime Agent Control Standard, an independent open specification (v0.1.0, May 2026) incorporated into the OWASP GenAI Security Project in September 2026 and currently at v0.1 public preview. This guide is intentionally broader across purpose, specification, testing, human oversight, change, recovery and governance.

Source family	Relationship to this guide
Vendor guidance	Strong practical patterns for agent building, guardrails, evaluation, security and human intervention, but normally platform- or implementation-oriented rather than a neutral control catalogue plus assessment intake.

Positioning discipline

The conclusion above is a current working assessment, not a claim of originality over every publication. The landscape should be rechecked before publication. If a mature equivalent emerges, map to it, reuse it or contribute rather than creating unnecessary competing terminology.

Appendix D — Dogfooding and Continuous Improvement

A control catalogue should be tested against real systems, not only reviewed conceptually. The objective is not to “pass” them. It is to test whether the framework identifies meaningful risks and controls, recognises controls already present, exposes unnecessary duplication, and produces remediation specific enough to drive engineering work.

What a useful dogfooding exercise should test

- Did the intake surface facts that materially changed the proposed controls?
- Did the catalogue identify real control gaps that the team had not already articulated?
- Did it recommend controls that are irrelevant or disproportionate — and can the framework explain why they should be rejected?
- Can existing controls be evidenced as implemented and operating, rather than merely described?
- Does the assessment distinguish defence in depth from wasteful duplication?
- Does the residual-risk statement help an accountable person make a real decision?
- Can a governance agent follow the same workflow and produce a traceable first-pass assessment without inventing missing facts?

Living-document rule

If a real use case exposes a material control mechanism, failure mode or classification the catalogue misses, add or revise the framework.

Appendix E — Version and Contribution Model

Version	Timing	Purpose
v0.4 — Author-reviewed redline	10 September 2026	Catalogue expanded from 101 to 113 controls with twelve new control cards; eight prose sections added, including the guide’s organising proposition and a per-family relationship to existing work; ten correction items from external review applied; control-metadata schema and spelling normalised; author attribution added to the title page.
v0.3 — Externally reviewed draft	8 September 2026	Agreed restructure: practical-use method moved before the catalogue; core proposition surfaced; dogfooding moved to the appendices; editorial and terminology cleanup. Appendix D, Dogfooding and Continuous Improvement, was added. The appendices formerly lettered D and E are now E and F. A reference to “Appendix D” or “Appendix E” made against an earlier edition therefore points to different content here.
v0.2 — Author review	4–8 September 2026	First author review: positioning and wording amendments, external-use neutralisation and structural review comments.
v0.1 — Working draft	3 September 2026	Initial 100-control baseline, assessment intake, matrices, standards alignment and machine-assisted assurance design.
v0.x — Living series	Ongoing	Add material new controls, failure modes, implementation patterns, evidence lessons, standards changes and case-study findings with version history.
v1.0 — Baseline	13 September 2026	Stable public release after expert review, practical validation, source verification and editorial/publishing pass.

Contribution principle: a proposed control or change should ideally state the failure mode it addresses, why the mechanism should work, implementation evidence, how it can fail, and where it differs from or complements existing catalogue controls. This keeps the catalogue from growing through synonyms and vague best practices.

Appendix F — First Dogfooding Assessment Template

Use this template to assess products, then reuse it for other applications, workflows and requirements. The aim is both system assurance and framework validation.

Step	Assessment activity
1. Scope	Product / component / requirement; intended purpose; environment; owners; version.
2. Intake	Complete Section 9, citing existing requirements, architecture, policies and evidence wherever possible.
3. Scenarios	List normal failure, misuse, adversarial, dependency and control-failure scenarios; rate inherent risk using the organisation's method.
4. Existing controls	Map implemented mechanisms to stable catalogue IDs; identify local implementation detail rather than assuming the catalogue definition proves existence.
5. Effectiveness	Rate design adequacy, implementation status, operating effectiveness and evidence confidence separately.
6. Gaps / overlaps	Identify missing controls, weak controls, unnecessary controls and correlated or duplicative controls.
7. Standards / policy	Map applicable obligations using Sections 18, 19 and 20 only as navigation; verify authoritative requirements and scope independently.
8. Residual risk	State residual risk, uncertainty, accepted assumptions and decisions requiring accountable human approval.
9. Remediation	Prioritise actions by risk reduction, dependency and implementation effort; link tasks/tests back to control IDs.
10. Framework feedback	Record missing controls, misleading metadata, poor recommendations, terminology problems and evidence patterns to improve the next catalogue version.

Dogfooding success questions

- Did the intake change the assessment materially, or was it merely administrative?
- Did the catalogue identify real gaps or over-controls that practitioners recognised as important?
- Did stable IDs make engineering, testing and governance conversations more precise?
- Could assessors show evidence that existing controls operate, rather than only point to policy text?
- Did the framework recommend genuine defence in depth rather than multiple controls with the same failure mode?
- Was the residual-risk output specific enough to support a real acceptance or remediation decision?
- Could a governance agent follow the process while clearly separating known facts from inference and unresolved questions?

Closing Note

The title question is intentionally simple. Getting agents to do what we want is not primarily about discovering a perfect instruction. It is about translating human intent into a layered system of specification, architecture, authority, validation, testing, oversight, monitoring, intervention and accountability — and then gathering evidence that those layers work in the environment where the agent actually operates.

This guide is meant to be used, challenged and changed. The most valuable additions should come from real deployments: a risk the catalogue did not expose, a control that failed differently than expected, a test that revealed correlated failure, a standards mapping that proved misleading, or a simpler mechanism that provided stronger assurance. Those findings are the route from a useful reference document to a mature control body of knowledge.