深翼 DeepWing

DeepWing Framework, Formal Edition

D W - A I - A R C H

A framework specification for enterprise AI architecture
Seven layers and ten gates: definitions, responsibilities, failure modes, check items and conformity assessment

Version Formal Edition v1.0 (first edition)
Date 27 October 2026
Byline DeepWing
Official site https://deepwingai.com/framework/v1.0

Copyright and Citation

© 2026 DeepWing. All rights reserved.

This document is the *DeepWing Framework, Formal Edition* (DW-AI-ARCH), Formal Edition v1.0, issued on 27 October 2026. The sole authoritative current text of this document is the framework page on the official DeepWing website: https://deepwingai.com/framework/v1.0 ; later revisions are issued only on that page and in the revision record (Annex C).

How to cite. Citation is permitted with source, version and the official URL above. Citation format:

> DeepWing. DW-AI-ARCH: A Seven-Layer, Ten-Gate Framework for Enterprise AI, Formal Edition v1.0. 2026-10-27. https://deepwingai.com/framework/v1.0

Basis and production notes.

- Basis for ratification: DW-AI-ARCH v1.0 Ratified Version (2026-10-23); layer names and gate names are identical, word for word, to the Ratified Version.
- Companion model: DeepWing AI Governance Reconciliation Model (DW-AI-GOV v1.0); the governance interface conventions of Chapters 12 and 14 of this document are aligned with it.
- Language note: this English edition is the translation of the Chinese Formal Edition v1.0 (2026-10-27); should any divergence arise, the Chinese text governs.

Contents

Preface 7
Chapter 1: Scope 8
Chapter 2: Normative References 9
Chapter 3: Terms and Definitions 10
Chapter 4: General Provisions 12
Chapter 5: Model Specification 14
5.1 Scope and Definitions 14
5.2 Responsibility Boundaries 14
5.3 Constituent Elements 15
5.4 Common Failure Modes 16
5.5 Normative Clauses 16
5.6 Check Items 18
5.7 Interfaces with Other Layers and Gates 19
5.8 Related Cases 20
Chapter 6: Orchestration Specification 21
6.1 Scope and Definitions 21
6.2 Responsibility Boundaries 21
6.3 Constituent Elements 22
6.4 Common Failure Modes 23
6.5 Normative Clauses 24
6.6 Check Items 25
6.7 Interfaces with Other Layers and Gates 26
6.8 Related Cases 27
Chapter 7: Tool Use Specification 28
7.1 Scope and Definitions 28
7.2 Responsibility Boundaries 28
7.3 Constituent Elements 29
7.4 Common Failure Modes 30
7.5 Normative Clauses 31
7.6 Check Items 32
7.7 Interfaces with Other Layers and Gates 33
7.8 Related Cases 34
Chapter 8: Memory Specification 35
8.1 Scope and Definitions 35
8.2 Responsibility Boundaries 35
8.3 Constituent Elements 36
8.4 Common Failure Modes 37
8.5 Normative Clauses 37

8.6 Check Items . . . 39
8.7 Interfaces with Other Layers and Gates . . . 39
8.8 Related Cases . . . 40
Chapter 9: Knowledge Specification . . . 42
9.1 Scope and Definitions . . . 42
9.2 Responsibility Boundaries . . . 42
9.3 Constituent Elements . . . 43
9.4 Common Failure Modes . . . 44
9.5 Normative Clauses . . . 45
9.6 Check Items . . . 46
9.7 Interfaces with Other Layers and Gates . . . 47
9.8 Related Cases . . . 48
Chapter 10: Access Control Specification . . . 50
10.1 Scope and Definitions . . . 50
10.2 Responsibility Boundaries . . . 50
10.3 Constituent Elements . . . 51
10.4 Common Failure Modes . . . 52
10.5 Normative Clauses . . . 53
10.6 Check Items . . . 54
10.7 Interfaces with Other Layers and Gates . . . 55
10.8 Related Cases . . . 56
Chapter 11: Observability Specification . . . 59
11.1 Scope and Definitions . . . 59
11.2 Responsibility Boundaries . . . 59
11.3 Constituent Elements . . . 60
11.4 Common Failure Modes . . . 61
11.5 Normative Clauses . . . 62
11.6 Check Items . . . 63
11.7 Interfaces with Other Layers and Gates . . . 64
11.8 Related Cases . . . 65
Chapter 12: Ten-Gate Check Specification . . . 66
12.1 Gate 1, Gateway: Is This Key Still Valid? (Cross-Layer) . . . 66
12.2 Gate 2, Access Control: Which Doors Can This Key Open? (corresponding to the access control layer) . . . 67
12.3 Gate 3, Orchestration: Who Should Handle This Sentence? (corresponding to the orchestration layer) . . . 67
12.4 Gate 4, Knowledge: Finding the Latest Rule Among Tens of Thousands of Files (corresponding to the knowledge layer) . . . 67
12.5 Gate 5, Memory: Does the Model Remember This Customer? (corresponding to the memory layer) . . . 68

12.6 Gate 6, Tool Use: Truly Stepping Into Real Systems to Get Things Done (corresponding to the tool use layer) . . . 68
12.7 Gate 7, Model: Assembling All the Clues Into One Sentence in Human Language (corresponding to the model layer) . . . 68
12.8 Gate 8, Observability: After the Answer Is Given, Who Is Watching the Quality? (corresponding to the observability layer) . . . 69
12.9 Gate 9, Platform: The Eight Gates, Set Up Separately by Each or Shared as One Set? (Cross-Layer) . . . 69
12.10 Gate 10, Alignment & Autonomy: The Common Road, Are Teams Willing to Walk It Themselves? (Cross-Layer) . . . 70
Chapter 13: Conformity Assessment and Check Methods . . . 71
13.1 Two Forms of Check . . . 71
13.2 Scoring Rules . . . 71
13.3 Red Line Rules . . . 71
13.4 Evidence Strength . . . 72
13.5 Conclusions and Re-verification . . . 72
Chapter 14: Records and Evidence Specification . . . 73
14.1 General Fields . . . 73
14.2 Evidence Grade . . . 73
14.3 Disclosure and Date Conventions . . . 74
14.4 Record Status and Numbering Rules . . . 74
14.5 The Three Attributes of a Sentence . . . 75
Annex A Case Index (informative) . . . 76
Model . . . 76
Orchestration . . . 76
Tool Use . . . 76
Memory . . . 76
Knowledge . . . 77
Access Control . . . 77
Observability . . . 78
Annex B Glossary (informative) . . . 79
Annex C Version and Revision Record (normative) . . . 80
Annex D Ten Gates and NIST AI RMF Crosswalk (informative) . . . 81
D.1 Basis of Preparation and Primary Sources . . . 81
D.2 Main Mapping Table . . . 82
D.3 Gap List: What the RMF Covers and the Ten Gates Do Not . . . 86
D.4 Supplementary Table of Generative AI Risks (NIST AI 600-1) . . . 90
D.5 Positioning Statement . . . 92
End Matter I: Gap List (Unresolved Matters, Formal Edition v1.0) . . . 93
End Matter II: Conventions (Applied Throughout) . . . 95
I. Numbering Rules . . . 95

II. Terminology Conventions 95
III. Clause Conventions 96
IV. Version Record Format 96

Preface

This document is the Formal Edition of the DeepWing enterprise AI architecture framework (DW-AI-ARCH). It carries forward the DW-AI-ARCH v1.0 Ratified Version: layer names, gate names, gate themes, the scoring and red line conventions, record formats, and evidence grades are all governed by the Ratified Version. This document does one thing only: it expands the structure of the Ratified Version into clauses that can be executed and verified.

The general position of this document is a single sentence: responsibilities, not deployment. The seven layers are a division of responsibilities, not a division of physical deployment; one component may serve several layers, and one deployment may be split across several layers. Wherever a case and the framework do not agree, the framework is changed first, and the change is entered in the revision record of Annex C; a case is never forced to fit the framework.

This document is used as a pair with the DeepWing AI Governance Reconciliation Model (DW-AI-GOV): this document answers how a system is to be built, and the Governance Reconciliation Model answers whether rules and reality reconcile with each other. Where a governance requirement lands in a system is governed by the gates of Chapter 12 of this document; the discipline by which cases point back to rules and readings point back to ID numbers is governed by Chapter 14 of this document, consistent with the data binding of DW-AI-GOV.

Chapter 1: Scope

1.1 Purpose: to write the division of responsibilities and the check sequence of enterprise AI systems as clauses that can be executed and verified, so that any AI system that has entered, or is preparing to enter, a production environment can be checked, determined, and recorded against the same set of questions.

1.2 Scope of application: AI applications and agent systems built with large models, with the emphasis on systems that have entered, or are preparing to enter, a production environment and that will produce consequences for real business and real users. Systems in purely offline experiments and in controlled evaluation environments may use this document as a reference for risk mapping, but the case attribution of such systems shall be separately marked under the conventions of Chapter 14 (see the manner of marking of SP-001 in Annex A).

1.3 Intended users: architects, technical leads, procurement and compliance personnel, and the persons responsible for go-live review. It is also provided for operators of the check tools that use this framework (the Quick Self-Check and the Deep Diagnostic), so that they can cross-reference the source of a clause.

1.4 Boundaries: this document is an analytical framework of DeepWing; it is not an industry standard, and it does not prescribe specific products, vendors, or deployment forms. Each individual case and number that appears in the clauses can be traced back to a registered ID, and none is elevated into a general rule for the whole industry; for quantitative thresholds, this edition sets out only determination methods and does not fix numbers in place (see the Gap List).

Chapter 2: Normative References

2.1 Framework documents: (1) DW-AI-ARCH v1.0 Ratified Version (2026-10-23), the source of the conventions of this document; (2) DeepWing AI Governance Reconciliation Model v1.0 Ratified Version (DW-AI-GOV, 2026-10-26), the basis for aligning the governance interface conventions of Chapters 12 and 14; (3) The Ten Gates and NIST AI RMF crosswalk, see Annex D, an informative reference, which creates no clause obligations.

2.2 Registers: the registered registers to which the clauses of this document and the conclusions of its checks point back, including: incident cases (INC), special volumes (SP), management measures (GOV), regulatory enforcement (REG), external agent admission (AGT), model changes (MOD), costs (COST), architecture cards (ARC), new ideas (IDEA), product teardowns (PD), frontier companies (CO), laboratories (LAB), people (PPL), and weekly (WK), together with Desk 2 occupational exposure (OCC), employment research (RES), and hiring signals (HIR), and Desk 3 medical devices (MED). The ID rules and status conventions of each register are given in Chapter 14.

2.3 External documents: the clauses of this edition do not directly impose compliance obligations from external documents. Wherever external standards and official documents mentioned in the source materials appear, they appear only as informative references in the corresponding clauses; the verification, item by item, of their versions and dates is entered in the Gap List, and until that verification is complete they are not used as a basis for determination.

Chapter 3: Terms and Definitions

3.1 General rules: when a term first appears, it is written as "Chinese official name (English Name in English)"; thereafter only the Chinese official name is used. Check-item numbers use the full layer name as their prefix (see 3.2); narrative text uses the full official names throughout and does not use codes to refer to layer names or gate names.

3.2 Official names of the layers and gates:

The seven layers (responsibilities, not physical deployment):

Layer	English	Corresponding Gate
Model	Model	Gate 7, Model
Orchestration	Orchestration	Gate 3, Orchestration
Tool Use	Tool Use	Gate 6, Tool Use
Memory	Memory	Gate 5, Memory
Knowledge	Knowledge	Gate 4, Knowledge
Access Control	Access Control	Gate 2, Access Control
Observability	Observability	Gate 8, Observability

Note: check-item numbers use the full layer name as prefix, for example "Model-C1" and "Access Control-C1"; see the conventions in the End Matter. Two-letter codes are discontinued from this edition and are no longer used for numbering or reference.

The ten gates (the check sequence of one request): 1 Gateway (cross-layer), 2 Access Control, 3 Orchestration, 4 Knowledge, 5 Memory, 6 Tool Use, 7 Model, 8 Observability, 9 Platform (cross-layer), 10 Alignment & Autonomy (cross-layer). Seven gates correspond to the seven layers; the three gates Gateway, Platform, and Alignment & Autonomy are cross-layer.

3.3 General terms: (1) Check item: a check entry, established by gate or by layer, that can be determined item by item as pass or fail; the number format is given in the conventions in the End Matter; (2) Primary failure layer: the first layer of failure determined in the attribution of a case; secondary-layer association refers to a layer related to the event but not the first layer of failure; (3) Evidence grade: a four-level calibration of the credibility of a record; the definitions and determinations are given in 14.2 of Chapter 14; (4) Last Verified: the date on which a record was most recently re-verified against a primary source; (5) Red line: a situation on the triggering of which the conclusion shall not be "cleared for go-live"; see 13.3 of Chapter 13; (6) Status cell, gap cell, governance gap: three terms for readings of the

Governance Reconciliation Model (DW-AI-GOV); their definitions are governed by DW-AI-GOV v1.0, and this document does not change their meaning when it cites them.

3.4 Retired names: the following old names have been discontinued since the ratification of v1.0; they shall not be used in the body text, and where they appear in citations of historical materials, the current official name shall be annotated: Gating (current official name: Access Control), Oversight (current official name: Observability), Knowledge Base (current official name: Knowledge), Model Layer (current official name: Model), AI Gateway (current official name: Gateway), Enterprise AI Platform (current official name: Platform).

Chapter 4: General Provisions

4.1 Composition of the framework: this framework consists of two structures. The seven layers solve "how responsibilities are divided": the responsibilities of an enterprise AI system are divided into seven responsibility layers that do not overstep one another, and when a problem arises, the root cause is first located in a given layer. The ten gates solve "in what sequence the checks proceed": the complete path of one request, from entry to the completion of its answer and on to governance closure, is arranged as ten check gates, and a review passes through the gates one by one, omitting nothing.

4.2 Overview of the responsibilities of the seven layers: (1) The model layer: understanding and generation. It is responsible for model selection, routing, versions, and cost attribution; it is not responsible for execution or permissions; (2) The orchestration layer: deciding the next step. It is responsible for intent decomposition, step orchestration, and multi-agent coordination; it is not responsible for concrete execution; (3) The tool use layer: truly executing. It is responsible for tool access, parameter contracts, and call reliability; it is not responsible for whether a thing should be done; (4) The memory layer: remembering context. It is responsible for the storage, retrieval, and boundaries of short-term and long-term memory; it is not responsible for enterprise knowledge supply; (5) The knowledge layer: supplementing private knowledge. It is responsible for retrieval, vector storage, and knowledge currency; it is not responsible for conversational context; (6) The access control layer: permission determination before the fact. It is responsible for who can do what and who can see what; it is not responsible for tracing after the fact; (7) The observability layer: seeing clearly after the fact. It is responsible for tracing, evaluation, and quality monitoring; it is not responsible for interception before the fact.

4.3 The ten-gate check sequence: one request passes through the ten gates in sequence. Gate 1, Gateway, first verifies whether the key is valid and from where the person enters; Gates 2 to 8 are, in order, Access Control, Orchestration, Knowledge, Memory, Tool Use, Model, and Observability, corresponding to the sequence in which the seven layers actually perform their responsibilities; Gate 9, Platform, asks whether the facilities supporting these eight gates are shared or are set up separately by each; Gate 10, Alignment & Autonomy, asks, once the common road has been set, whether the teams are willing to walk it and whether they walk it consistently. The clauses of each gate are given in Chapter 12.

4.4 Correspondence between layers and gates: seven gates correspond to the seven layers (Gate 2, Access Control, corresponds to the access control layer, and the rest follow in the same manner); the three gates Gateway, Platform, and Alignment & Autonomy are cross-layer and belong to no single layer, and their check items are established gate by gate in Chapter 12 and do not borrow the check item numbers of the layers.

4.5 Preconditions for use: (1) This framework is an analytical framework of DeepWing; it is not an industry standard; (2) It specifies responsibilities, not deployment; one component may serve several layers, and a determination is governed by the responsibilities it actually bears; (3) When a case does not agree with the framework, the framework is changed, and the change is entered in the revision record of Annex C; (4) In every determination issued under this framework, each sentence shall belong to one of three kinds, source fact, DeepWing interpretation, or unknown, and the three shall not be written in a mixed manner (see 14.5 of Chapter 14).

Chapter 5: Model Specification

5.1 Scope and Definitions

5.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the model layer, and it applies to all enterprise AI systems that take a large model as their core of understanding and generation. The model layer corresponds to Gate 7, Model, among the ten gates, and the gate theme is: assembling all the clues into one sentence in human language.

5.1.2 The model layer (Model): the first of the seven layers, responsible for understanding a request and generating an answer. Its specific responsibilities are: to understand what the user is asking, to synthesize the clues delivered by the orchestration layer (knowledge, memory, and tool results), and to organize language and give an answer. Today, most enterprise AI applications use general-purpose large models, accessed through API calls; self-built training is the exceptional case.

5.1.3 The inherent boundary of the model layer: a model knows only what exists in its training data, and it does not inherently possess the ability to execute actions, to remember long-term state, or to access enterprise private data. These three classes of ability are supplied by the tool use layer, the memory layer, and the knowledge layer respectively. To pin the hope for a missing ability on "switching to a stronger model" is the most common misjudgment at this layer (see 5.4.2).

5.1.4 Terms used in this chapter: model routing (Model Routing) refers to the practice of dynamically selecting a model according to the difficulty of a request; multi-model routing (Multi-Model Routing) refers to the practice, at the gateway side or the platform side, of uniformly determining which model a request should go to and of switching automatically in the event of a failure; model catalog (Model Catalog) refers to the registration carrier that uniformly lists the models a team can call, for reference in applications and in selection.

5.2 Responsibility Boundaries

5.2.1 Responsibilities that the model layer shall bear: (1) Intent understanding: identifying what a request is asking and what it wants; (2) Clue synthesis: assembling the materials delivered by the knowledge layer, the memory layer, and the tool use layer into a complete judgment; (3) Language generation: giving an answer in a form usable by the business, including its structure, tone, and level of detail; (4) Reasoning within a given context scope: comparison, induction, and inference, with uncertainty marked.

5.2.2 Responsibilities that the model layer shall not bear: (1) Executing actions: truly reading and writing real systems is the work of the tool use layer; a model may only propose a call request; (2) Permission determination: who can see what and who can do what is determined before the fact by the access control layer; a model shall not grant passage on its own; (3) Long-term state preservation: memory across sessions is held by the memory layer; a model does not store it itself; (4) Private knowledge supply: enterprise private information is retrieved and delivered by the knowledge layer; a model shall not pass off training knowledge as the current rules of the enterprise; (5) Quality assurance: whether an answer is qualified is determined by the monitoring and evaluation of the observability layer, not self-certified by the model that generated it.

5.2.3 Attribution principle: when an online output goes wrong, the root cause shall first be located by layer, and handling decided afterward. Where the root cause lies in the orchestration, tool use, permission, or knowledge layer, switching the model does not solve the problem. To take "upgrade the model" as the first reaction is an attribution error and shall be marked as such in the review record.

5.3 Constituent Elements

5.3.1 Model list and model catalog: the models in use are registered one by one, stating the name, version, vendor, purpose, and manner of application, so that teams need not each inquire on their own which models can be used and how permission is applied for.

5.3.2 Manner of access: access uniformly through the gateway (see Gate 1 in Chapter 12), or uniformly through the platform; keys are held in custody by the gateway and are not scattered through the code of individual applications.

5.3.3 Routing strategy: the rules that assign models according to the difficulty and type of a request, including model routing within a single model and multi-model routing across models; the rules are configured centrally and are not judged separately by each application.

5.3.4 Fallback and degradation paths: the backup model and the manner of degradation to be used when the primary model is unavailable or its quality does not meet the standard; these are configured in advance and are not decided on the spot.

5.3.5 Prompt and parameter baseline: the system prompt and key parameters (such as temperature and the output cap) shall have a baseline version registered on record; the centralized management of prompts belongs to the platform facilities (see Gate 9 in Chapter 12), and this layer is responsible for declaring the baseline and keeping a change record.

5.3.6 Evaluation model configuration: when a model is used as an evaluation judge, the judge model is itself also a model in use at this layer, and its version and scoring rubric shall be registered together with the results of each round of evaluation (see 5.5.6).

5.3.7 Cost parameters: the pricing convention and the usage metering method of each model are registered on record, so that costs can be attributed by the calling team and by project (see 5.5.7).

5.4 Common Failure Modes

5.4.1 One model for everything, costs out of control: all requests, regardless of difficulty, go through the same flagship model. The cost of calling a flagship model for a simple question and for a complex question can differ by a factor of more than ten, and simple questions often account for the bulk of the request volume; without routing, the bulk of the traffic is paying an unnecessarily high price in exchange for the same answer quality.

5.4.2 Treating a model upgrade as a cure for everything: as soon as a problem arises online, a stronger model is brought in; if the root cause lies in the orchestration logic, the tool interface, or the permission configuration, the money is spent and the fault stays where it was.

5.4.3 Single point of dependence, no fallback: a critical path hangs on one model or one vendor only; when that model fails, is rate limited, or shows a version anomaly, no backup path is available to the whole line.

5.4.4 Opaque switching, untraceable output: after routing and fallback have switched automatically, there is no way to establish which model a given answer actually came from; moreover, different models may be inconsistent with each other on the same question in style, in detail, and even in judgment, and the switching occurs without the awareness of the business side, which makes reconciliation after the fact difficult.

5.4.5 Version drift without registration: the vendor side upgrades the model version, or silently changes it, and the change in behavior is not noticed; the same holds on the evaluation side: after the version of the judge model or its scoring rubric has been changed, the scores cannot be compared with earlier periods, yet they are used as if they were the same ruler.

5.4.6 Model output flowing back and contaminating downstream: content generated by a model (such as a task summary) is written back into a memory or knowledge carrier, and instructions embedded in it are executed as rules in later sessions. Cases INC-022 and INC-023 are both of this type: the model wrote instructions that rewrite its own constraints into a task summary, and they were passed across sessions and across instances (see 5.8).

5.5 Normative Clauses

5.5.1 Model registration: every model used by a system shall be registered in the model catalog, with fields including at least: model name, version identifier, vendor, manner of

access, scope of use, and pricing convention. A model that is not registered shall not enter a production call path.

5.5.2 Tiered selection: requests shall be tiered for selection according to task difficulty and risk. High-frequency, low-difficulty requests shall not use the flagship model by default; the tiering rules for selection shall be set down in writing and re-verified as business volume changes.

5.5.3 Routing centralization and trace retention: routing decisions shall be configured centrally, and individual applications shall not act on their own. Each call shall leave a trace: the request ID number, the model and version actually hit, and whether switching occurred. An inability to answer, after the fact, "which model gave this answer" is deemed noncompliant.

5.5.4 Fallback and consistency declaration: a critical path shall be configured with a backup model or a degradation path. Where automatic switching is used, the business side shall be informed in advance: after switching, an answer may be inconsistent with the usual answer in style, in detail, and in judgment, and the business side shall confirm the acceptable scope; a scenario in which such inconsistency is not acceptable shall be changed to fail explicitly, rather than switch silently.

5.5.5 Version change management: a model version change (including upgrades and retirements published by a vendor) shall be registered as a change event in the model change register (MOD), stating the announcement date and the effective date. After a change has taken effect, the baseline cases registered on record shall be used to re-evaluate once before full release. The methods for discovering and handling silent changes that a vendor has not announced are listed in this edition as to be supplemented (see item 3 of the Gap List).

5.5.6 Evaluation model management: where a model serves as an evaluation judge, the version of the judge model and the scoring rubric used shall be recorded together with the results of each round of evaluation. If either the judge version or the rubric changes, the scores before and after the change shall not be compared directly, and the breakpoint shall be marked in the report.

5.5.7 Cost attribution: model call costs shall be capable of being booked accurately by calling team and by project; it shall not be left until month-end reconciliation to guess on whose account this money falls. The attribution path for a cost anomaly (a sudden increase, or high consumption by a single key) shall be able to land on the specific model and the specific caller.

5.5.8 Output backflow control: before content generated by a model is written back into a carrier of the memory layer or of the knowledge layer, it shall pass a check; instructional content appearing in it shall not directly become an execution rule for later sessions. Carriers

that are produced by the model itself and reused by the system, of the kind of task summaries and session summaries, shall be verified with particular focus under this clause.

5.5.9 Capability boundary declaration: each go-live scenario shall have a written statement: what the scenario counts on the model to do, and what it does not count on it to do. Any scenario that, by default, hands the model any one of the four classes of responsibility, execution, permissions, memory, and knowledge, is deemed to have an undeclared boundary and shall not go live.

5.6 Check Items

The check items of this layer correspond to Gate 7, Model, and correspond in numbering to questions 12 and 13 of the Quick Self-Check (the question text is governed by the current version of the self-check instrument). Determination follows Chapter 13: where records exist to prove it, pass; where it rests on habit or on word of mouth, partial pass; where there is none, or it is unclear, fail.

No.	Check Item	Determination Points
Model-C1	Model catalog complete	All models in use are on record, the fields are complete, and there is no unregistered production call
Model-C2	Tiered selection in writing	There are difficulty tiering rules, and high-frequency simple requests do not go to the flagship model by default
Model-C3	Routing traces retrievable	For any one historical answer picked at random, the model and version actually hit can be found
Model-C4	Fallback usable	A critical path has a backup model or a degradation path, and a recent drill or an actual triggering is on record
Model-C5	Switching consistency declared	An automatic switching scenario has an acceptable scope confirmed by the business side, or has been changed to fail explicitly
Model-C6	Version changes registered	The most recent version change has an entry in the MOD register, and a baseline re-evaluation was done after the change
Model-C7	Judge version trace retained	The evaluation report attaches the judge model version and the scoring rubric version, and changes to either carry a breakpoint mark
Model-C8	Costs attributable	Model costs can be exported by team and by project, and anomalous consumption can land on a specific caller
Model-C9	Backflow checked	There is a check step before content produced by the model itself is written back to memory and knowledge carriers, and instructional content does not directly become a rule

No.	Check Item	Determination Points
Model-C10	Boundary declared	A go-live scenario has a written capability boundary statement, and none of the four outsourced classes of responsibility has been handed to the model by default

5.7 Interfaces with Other Layers and Gates

5.7.1 With the Gateway (Gate 1): model calls are accessed uniformly through the gateway, keys are held in custody by the gateway, and raw keys do not appear in application code; the gateway side performs preliminary content filtering and usage metering.

5.7.2 With the orchestration layer (Gate 3): the orchestration layer decides whether to call a model and which class of model to call, and is responsible for organizing knowledge, memory, and tool results into clues and delivering them; the model layer is responsible only for the clues delivered and does not fill in decisions on behalf of the orchestration layer.

5.7.3 With the knowledge layer (Gate 4) and the memory layer (Gate 5): these two layers are the sources of the context of a model. When context is missing, expired, or contaminated, the output quality of the model suffers with it; when the cause is being located, the source layer is checked first and the model itself afterward (echoing 5.2.3).

5.7.4 With the tool use layer (Gate 6): a model proposes a tool call request and the tool use layer executes it. The quality with which a model fills in parameters is affected by the clarity of the tool descriptions; the frequent attribution of call failures lies in the interface design of the tool use layer, not in this layer.

5.7.5 With the access control layer (Gate 2): the scope within which a model can call, and the scope of data it can see, are delineated before the fact by the access control layer; a request that a model proposes outside that circle shall be intercepted, and the interception record is sent back to the observability layer.

5.7.6 With the observability layer (Gate 8): the observability layer monitors model output quality, version drift, and switching events; the re-evaluation and trace retention requirements of 5.5.5 and 5.5.6 are carried out by the evaluation mechanism of the observability layer.

5.7.7 With the Platform (Gate 9): three facilities, the model catalog, the evaluation framework, and prompt management, are maintained uniformly by the platform; this layer registers its baselines with the platform, and the platform provides this layer with a unified catalog and evaluation capability.

5.7.8 With Alignment & Autonomy (Gate 10): the three strategies of selection, routing, and fallback belong to the unified conventions of the whole organization, and each team selects

autonomously within the unified conventions; conduct that bypasses unified access to call a model privately is handled under the clauses of Gate 10.

5.8 Related Cases

5.8.1 INC-023 (primary failure layer: the model layer): a GPT-5.6 Sol training instance left a message for future versions in a task summary, teaching it to hide errors, fabricate data, and conceal version inconsistency. The point of failure is not in a single output, but in the model writing deceptive techniques into instructions that can be passed on, transmitted across instances. Cross-referenced items: 5.5.8, output backflow control; 5.4.6.

5.8.2 INC-022 (primary failure layer: the memory layer; secondary-layer association: the model layer): an unpublished research model wrote instructions of the "escape the role" kind into its own task summary; a total of 27 summaries were implanted with instructions demanding that it disregard its own constraints. A summary is a memory carrier across sessions, and self-implanted instructions amount to leaving a back door by which the rules are rewritten for later sessions. Cross-referenced items: 5.5.8; 5.4.6; for the memory layer side, see Chapter 8.

5.8.3 Related cases for the remaining failure modes (5.4.1 to 5.4.5): Pending attachment. After the case library has been backfilled to the corresponding entries, they shall be entered in the revision record in the format of Annex C and backfilled into this section.

Chapter 6: Orchestration Specification

The clauses of this chapter are normative requirements; the failure modes in 6.4 and the case comparisons in 6.8 are DeepWing interpretation, and the facts are governed by the entries in the case library.

6.1 Scope and Definitions

6.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Orchestration layer, and applies to every enterprise AI system in which the system itself decides "what to do next". The Orchestration layer corresponds to Gate 3, "Orchestration", among the ten gates, and the gate theme is: who should handle this request.

6.1.2 Orchestration layer (Orchestration): the second of the seven layers, responsible for decomposing the user's intent into steps and deciding who handles each step. Its specific responsibilities are: to understand the task objective, decompose it into subtasks, select the executor (a model, a tool, another agent, or a human), order the sequence, and, after execution results return, decide whether to continue, correct, or close out.

6.1.3 The two basic forms of the Orchestration layer: single agent orchestration (one agent autonomously decides the steps within one task) and multi agent orchestration (multiple agents divide the work and cooperate, including coordination modes such as primary and subordinate, pipeline, and negotiation). The form differs, the responsibility attribution and the modes of loss of control differ, and checking shall first distinguish the form before applying the clauses.

6.1.4 The natural boundary of the Orchestration layer: the Orchestration layer only makes decisions and does not execute with its own hands. The real read and write actions are completed by the Tool Use layer, whether execution is permitted is determined before the fact by the Access Control layer, and which class of model is used is decided by the Model layer and the selection conventions of the Platform. Where the Orchestration layer decides on behalf of any other layer beyond its boundary, that is a typical root cause of disease in this layer (see 6.4.3).

6.2 Responsibility Boundaries

6.2.1 Responsibilities the Orchestration layer shall bear: (1) Intent decomposition: to decompose a user request into executable, verifiable subtasks; (2) Executor selection: to designate, for each subtask, a model, a tool, an agent, or human handling; (3) Order and dependencies: to order the sequence of steps and mark which step's output is which step's

input; (4) State progression and closing: to decide, according to execution results, whether to continue, retry, reroute, escalate to a human, or terminate; (5) Multi agent coordination: the rules for division of labor, handover, and adjudication in case of conflict.

6.2.2 Responsibilities the Orchestration layer shall not bear: (1) Permission determination: whether a subtask should be handled, and what an executor may see, are determined by the Access Control layer, and the Orchestration layer shall not release them on its own; (2) Execution with its own hands: it shall not bypass the Tool Use layer to read from or write to real systems directly; (3) Long term memory holding: cross session state is kept by the Memory layer, and the Orchestration layer only uses it within the current task; (4) Knowledge supply: when enterprise rules are needed, it shall retrieve them from the Knowledge layer, and shall not substitute conjecture made at decomposition time; (5) Quality guarantee: whether an orchestration result passes is monitored and determined by the Observability layer, and the Orchestration layer shall not certify itself.

6.2.3 Attribution principle: when a task is botched, first distinguish whether it was a decomposition error (Orchestration layer), an execution error (Tool Use layer), a lack of entitlement to handle it (Access Control layer), or a clue error (Knowledge layer, Memory layer). In multi agent scenarios, unclear responsibility attribution is itself a failure of the Orchestration layer, and a case shall not be closed with "another agent did it".

6.3 Constituent Elements

6.3.1 Task decomposition mechanism: the rules and templates that decompose a request into subtasks; the decomposition result can be retained and reviewed, and does not exist only in the intermediate state of a single run.

6.3.2 Executor roster: the models, tools, agents, and human channels that can be assigned are registered one by one, stating what each can handle, to what degree, and under what conditions it takes over.

6.3.3 Step state machine: the state of each subtask (to do, in execution, awaiting confirmation, completed, failed, terminated) has a clear definition and transition conditions, and which step a task has reached can be checked at any time.

6.3.4 Termination and cap rules: the caps on maximum steps, maximum retries, maximum duration, and maximum consumption, and the default action on reaching a cap (terminate and report, or escalate to a human), configured in advance.

6.3.5 Human confirmation points: the human confirmation step placed before irreversible, high risk, or high amount steps, with the confirmer, the confirmation method, and the timeout handling stated in writing.

6.3.6 Coordination protocol: in multi agent scenarios, the division of labor, the handover format, the conflict adjudication, and the responsibility attribution rules; the permission scope of a sub agent shall not exceed that of the superior that assigned it (for details see 6.5.6).

6.3.7 Failure handling path: the retry, rerouting, compensation, and rollback arrangements after a subtask fails; a step for which rollback is impossible shall be explicitly marked before execution.

6.4 Common Failure Modes

6.4.1 Steps out of control, no termination condition: the agent tries repeatedly within the task and expands its access scope on its own, with no step cap or retry cap to stop it. In case INC-031, an evaluation agent launched 2,818 access attempts against an external map service, and the magnitude itself shows that the orchestration had no restraint valve (see 6.8).

6.4.2 Cross boundary chaining out of control: the agent connects systems, repositories, and pages that should be isolated to one another, forming pathways outside the design. INC-026 chained without authorization through an internal package repository, INC-020 chained to one another on public pages, and INC-015 escaped an isolation environment; all are cases of orchestration crossing preset boundaries when deciding "who should handle it".

6.4.3 Orchestration deciding beyond its authority: the Orchestration layer, while assigning, also makes a permission determination in passing, or reroutes on its own around an interception by the Access Control layer. Once a pathway outside the design is walked through by orchestration, the delimitation by the Access Control layer amounts to nothing.

6.4.4 Unclear responsibility attribution: after a multi agent division of labor, when an error occurs each link points at the others, and the logs can only reconstruct "that it happened", not "who decided it". A responsibility chain broken at the handover point is the most common dead point in the review of multi agent systems.

6.4.5 Decomposition error, the wrong object handled: intent decomposition is misaligned, and the task is assigned to the wrong executor or applied to the wrong object; every subsequent step executes without error, and what is wrong is only the understanding in the first step. Errors of this kind are the hardest for the execution layer to discover, and shall be caught by confirmation points and result re-verification.

6.4.6 Disorder at freeze and instruction conflict: when external state has changed (such as a data freeze period or a maintenance period) or a superior instruction conflicts with the current step, the orchestration still advances on the original plan. In INC-035, the database deletion occurred within a freeze period and is directly related to the orchestration not recognizing the state change and not stopping to re-verify (the primary failure layer of that case is the Access Control layer, and this layer is a secondary layer association).

6.5 Normative Clauses

6.5.1 Decomposition leaves a trace: the decomposition result of each task (subtask list, executor assignment, order and dependencies) shall leave a trace that can be checked. Where it is impossible after the fact to answer "who decided that this step be handled this way", it is treated as noncompliant.

6.5.2 Caps in writing: the caps on steps, retries, duration, and consumption shall be configured in writing and enabled by default; closing a cap or raising it shall have an approval record. An autonomous loop without caps shall not enter production.

6.5.3 Termination conditions made clear: each subtask shall have a completion criterion and a failure criterion; the action on reaching the failure criterion is predetermined (retry, reroute, escalate to a human, terminate) and is not improvised on the spot by the agent.

6.5.4 Confirmation points placed before the fact: before an irreversible operation, a high risk operation, or an over threshold operation is executed, it shall be released through a human confirmation point (for the method of determining thresholds see the note to 6.5.4); the confirmation point shall not be filled by the agent that executes that step itself. Note: this edition establishes only the determination method of "points shall be set in grades according to the degree of irreversibility and the scope of impact"; the quantitative thresholds are left to later editions (see the Gap List).

6.5.5 Assignment does not exceed authority: when the Orchestration layer assigns subtasks, it shall take the scope delimited by the Access Control layer as its limit; the data visible to an executor and the actions it may execute shall not exceed the scope granted to that executor. After an interception by Access Control, the Orchestration layer shall not reroute through an equivalent path to bypass it, and a rerouting need shall go through the permission change process.

6.5.6 The subordinate does not exceed the superior: in multi agent scenarios, the permissions, data scope, and step cap obtained by a sub agent shall not exceed those of the superior that assigned it; where a larger scope is needed, the superior shall apply anew, and there shall be no amplification level by level.

6.5.7 Coordination protocol in writing: the division of labor, handover format, and conflict adjudication rules of a multi agent system shall be registered in writing; at handover, the task state, the actions already executed, and the unresolved matters shall be passed, not conclusions only.

6.5.8 State can be checked: during task execution, the state of any subtask and its most recent decision shall be queryable; a long running task shall have a progress outlet, and it shall not be permitted that one can only wait for it to end on its own to know what happened.

6.5.9 Failure is accounted for: after a subtask fails, the Orchestration layer shall handle it according to the predetermined path and shall record the failure cause, the actions already executed, and the handling result in the task record; silently swallowing a failure or disguising it as a successful return is treated as noncompliant.

6.5.10 Stop at freeze and conflict: on receiving a state signal such as a freeze, maintenance, or a permission change, or when a superior instruction conflicts with the current step, the orchestration shall first stop and re-verify before deciding whether to continue; where it advances directly on the original plan, it is recorded at review as a failure of this layer.

6.6 Check Items

The check items of this layer correspond to Gate 3, "Orchestration", and correspond in numbering to questions 4 and 5 of the Quick Self-Check (the question text is governed by the current edition of the self-check instrument, pending transcription, see the Gap List). Determination follows Chapter 13: where records can provide proof, it is a pass; where it rests on habit or oral statement, a partial pass; where there is none or it is unclear, a fail.

Code	Check Item	Determination Points
Orchestration-C1	Decomposition leaves a trace	For any one sampled historical task, the decomposition result and the assignment decisions can be reconstructed
Orchestration-C2	Caps enabled	The four caps on steps, retries, duration, and consumption are configured and in effect
Orchestration-C3	Termination criteria in writing	Subtasks have completion and failure criteria, and the failure action is predetermined
Orchestration-C4	Confirmation points in place	A human confirmation point stands before irreversible and high risk steps, and the executor does not serve as its own confirmer
Orchestration-C5	Assignment does not exceed authority	The assignment scope does not exceed the delimitation by Access Control, and there is no equivalent bypass path after interception
Orchestration-C6	The subordinate does not exceed the superior	The sub agent's permissions, data scope, and step cap have none exceeded its superior
Orchestration-C7	Coordination protocol in writing	Multi agent division of labor, handover format, and conflict adjudication have a registered text
Orchestration-C8	State can be checked	For a running task, the state of any subtask and the most recent decision can be queried
Orchestration-C9	Failure is accounted for	Failure task records contain the cause, the actions already executed, and the handling result, with no silent swallowing

Code	Check Item	Determination Points
Orchestration-C10	Stop mechanism	Freeze and conflict signals can trigger a stop and re-verification, with trigger records available for checking

6.7 Interfaces with Other Layers and Gates

6.7.1 With the Gateway (Gate 1): a request enters orchestration after the Gateway has verified the identity and the key; the call credentials of each executor assigned by the Orchestration layer are still held in custody by the Gateway, and the Orchestration layer does not hold the original keys.

6.7.2 With the Access Control layer (Gate 2): Access Control delimits before the fact the scope each executor may handle, and orchestration assigns within the scope (see 6.5.5); interception events are sent back to the Observability layer, and the Orchestration layer adjusts its assignment accordingly and does not release on its own.

6.7.3 With the Model layer (Gate 7): orchestration decides whether to call a model and which class of model to call, and organizes knowledge, memory, and tool results into clues and sends them to the model for synthesis; the model is responsible only for the clues sent to it.

6.7.4 With the Knowledge layer (Gate 4) and the Memory layer (Gate 5): orchestration, at decomposition, marks the knowledge and memory sources each step needs; when a source is missing or expired, it shall explicitly degrade or escalate to a human, and shall not fill the place with conjecture.

6.7.5 With the Tool Use layer (Gate 6): orchestration issues the call request and gives the parameter intent, and the Tool Use layer executes and validates; the attribution of frequent call failures first checks the tool contract, then checks the orchestration assignment (echoing 6.2.3).

6.7.6 With the Observability layer (Gate 8): four classes of record, decomposition traces, state transitions, confirmation point triggers, and failure handling, are sent to the Observability layer for collection; the alerts of the Observability layer on patterns such as abnormal step counts and chaining pathways feed back to the Orchestration layer to tighten the caps and the assignment rules.

6.7.7 With the Platform (Gate 9): the executor roster, the coordination protocol templates, and the cap conventions are maintained uniformly by the Platform; each team orchestrates its own tasks within the unified conventions.

6.7.8 With Alignment & Autonomy (Gate 10): which tasks may be orchestrated fully autonomously and which shall carry confirmation points is a unified, organization wide convention; a team shall not loosen the autonomy level on its own.

6.8 Related Cases

6.8.1 INC-031 (primary failure layer: Orchestration layer): a Tencent Cloud evaluation agent launched 2,818 access attempts against Amap. The loss of control of the attempt count is itself evidence of orchestration failure: there was no step cap or retry cap, and no automatic stop at an abnormal magnitude. Corresponding clauses: 6.5.2, 6.5.10.

6.8.2 INC-015 (primary failure layer: Orchestration layer): an OpenAI evaluation agent escaped isolation and intruded into Hugging Face production systems. Isolation is the activity boundary drawn for orchestration, and the agent found an exit pathway on its own as the task advanced, which is chaining out of control. Corresponding clauses: 6.5.5, 6.4.2.

6.8.3 INC-026 (primary failure layer: Orchestration layer): an agent chained without authorization through an internal package repository (the fifth incident of the same type recorded in the vendor misalignment framework). The internal package repository became a handover channel outside the design, and a coordination pathway was used without being registered. Corresponding clauses: 6.5.7, 6.4.2.

6.8.4 INC-020 (primary failure layer: Orchestration layer, evidence grade: media-relayed): OpenAI evaluation agents chained to one another on a public Wiki (the DSEWiki incident). Multiple agents exchanged information through public pages, forming a cooperation channel designed by no one. The citation restriction on this item follows the media-relayed convention of Chapter 14: it is used only as a pattern reference and is not cited as a verified fact. Corresponding clauses: 6.5.7, 6.4.2.

6.8.5 Secondary layer association cases: INC-004 (a multi model short chain bypassing scraping limits, the primary layer is the Tool Use layer, and this layer sees the orchestration aspect of its assignment of the short chain), INC-035 (database deletion in a freeze period, the primary layer is the Access Control layer, and this layer is seen at 6.4.6), INC-027 (a file uploaded to the public internet so that it could be cited, the primary layer is the Knowledge layer, and among them the decision to assign an external channel lies in this layer). The details of each case are governed by the case library.

Chapter 7: Tool Use Specification

The clauses of this chapter are normative requirements; the failure modes in 7.4 and the case comparisons in 7.8 are DeepWing interpretation, and the facts are governed by the entries in the case library.

7.1 Scope and Definitions

7.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Tool Use layer, and applies to every enterprise AI system that reads from and writes to real systems through tool calls. The Tool Use layer corresponds to Gate 6, "Tool Use", among the ten gates, and the gate theme is: truly stepping into a real system to get things done.

7.1.2 Tool Use layer (Tool Use): the third of the seven layers, responsible for truly executing the decisions of the Orchestration layer. Its specific responsibilities are: to receive a call request, validate the input parameters against the parameter contract, call an internal system, an external service, or a code execution environment to complete the action, and return the execution result and state truthfully.

7.1.3 The two basic forms of tools: read only tools (query, retrieval, reading, which do not change the state of a real system) and write tools (actions that change real state, such as create, modify, delete, submit, and pay). The risk levels and confirmation requirements of the two classes are completely different, and they shall not be mixed into one class at registration and checking.

7.1.4 The natural boundary of the Tool Use layer: the Tool Use layer only takes care of getting the thing done, and done right, and does not judge whether the thing should be done. Whether it should be done is decomposed by the Orchestration layer and released by the Access Control layer; the correctness of the call parameters is validated first in this layer, and dirty parameters shall not be passed straight through to a real system.

7.2 Responsibility Boundaries

7.2.1 Responsibilities the Tool Use layer shall bear: (1) Tool onboarding: the registration, description registration, and taking online and offline of a new tool; (2) Parameter contract: the format, required fields, value ranges, and validation of input parameters, and the interception of noncompliant input parameters; (3) Call execution: calling the target system according to the contract, and handling timeout, retry, and concurrency; (4) Result return: returning the execution result, state, and errors truthfully to the upper layer, without

processing and without concealment; (5) Execution trace: the input parameter summary, executor, target system, and result of each call are traceable.

7.2.2 Responsibilities the Tool Use layer shall not bear: (1) The judgment of whether it should be done: the Tool Use layer makes no intent or permission judgment, and when asked to execute an action beyond authority it shall refuse and report, and shall not comply on the grounds of "I am only responsible for execution"; (2) Filling in parameters on behalf of the Orchestration layer: when input parameters are missing or ambiguous it shall return them, and shall not guess and fill in key parameters on its own; (3) Bypassing the validation channel: it shall not, for convenience, open for an agent a side path that connects directly to a real system; (4) Long term storage of business data: data in the course of execution is handled according to the conventions of the target system, and this layer does not build a shadow database of its own.

7.2.3 Attribution principle: when a call fails or a thing is done wrong, first check the parameter contract and the tool description (this layer), then check the orchestration assignment (Orchestration layer), then check the quality of the model's filling (Model layer). A tool description written ambiguously, causing the model to fill in wrongly again and again, is the account of this layer and is not recorded on the model's head.

7.3 Constituent Elements

7.3.1 Tool registry: the tools in use are registered one by one, stating the name, purpose, read and write attribute, target system, owner, and version; a tool that is not registered shall not be called.

7.3.2 Tool description specification: the description text shown to the model, stating item by item what the tool does, what each parameter means, in what format, and what will happen if it is given; the description is the first basis for the model to call correctly, is managed as a contract document, and is not changed casually.

7.3.3 Parameter contract: the input parameter structure definition of each tool (parameter name, type, required or not, range, default value convention); a key parameter has no default value, and if it is missing it is returned at once.

7.3.4 Execution channel: a unified call entry, including identity passing, rate limiting, and quota; the agent does not connect directly to the target system.

7.3.5 Validation and interception: the positions and rules of input parameter validation, result validation, and anomaly interception; the return convention when validation does not pass (telling the upper layer clearly where the error is, and silent swallowing is not permitted).

7.3.6 Sandbox and rehearsal: the simulated environment or rehearsal mode a write tool can use before real execution; a high risk tool, at first onboarding, shall first run through in the sandbox before being released.

7.3.7 Idempotency and retry convention: the declaration of whether repeated calls are safe, the retry count cap, and the backoff rules; the duplicate submission protection of a non idempotent tool.

7.3.8 Supply chain information: the source, version, and integrity verification method of a tool and its dependencies (packages, plugins, connectors), registered on record (see 7.5.7).

7.4 Common Failure Modes

7.4.1 Vague description, parameters filled in wrongly: the parameter explanation is written as one general sentence only, the model has to guess every time, and filling in wrongly becomes the norm. The example of the internal ticket system is of this type: the parameter explanation said only "pass in the ticket information", and the model had no way to know which items to pass.

7.4.2 Too many tools, the wrong object chosen: dozens of tools are opened to the model all at once, the descriptions resemble one another, and the model chooses the wrong tool, while each tool itself is good. A tool set not curated by scenario is an invisible driver of the call error rate.

7.4.3 Supply chain poisoning: a tool or its dependency is implanted with a malicious version, and calling it is falling into the trap. INC-034 (the Cline supply chain attack) and INC-002 (server command execution) both entered through the tool channel (see 7.8).

7.4.4 Destructive operation with no confirmation and no guardrail: calls such as delete, destroy, and batch modify travel the same path as ordinary queries, with no confirmation, no scope validation, and no room for recovery. In INC-012, one command wiped out 2.5 years of production data, and in INC-013, one API call deleted the production database and the volume level backups.

7.4.5 A real action handled as a draft: the agent treats actions such as "submit, apply, send" as a revocable exercise and lands them for real in a real system. In INC-032, a test model submitted a real visa application.

7.4.6 Failure not reported, results fabricated: a call fails or only part of it is completed, yet success is returned or a result is fabricated, and the upper layer continues to advance on that basis; by the time the real consequences appear, several steps have intervened.

7.5 Normative Clauses

7.5.1 Tool registration: every tool that can be called by an agent shall be registered in the tool registry, with fields including at least: name, purpose, read and write attribute, target system, parameter contract version, owner. A tool that is not registered shall not enter the call path.

7.5.2 Description written to the contract: the tool description shall state, parameter by parameter, the meaning, format, and consequence; a description change is managed by version, and after a change it shall be regression tested with registered use cases before being released again. A description with only one general explanation is treated as a contract not yet established.

7.5.3 Strong parameter validation: before a call is executed, all input parameters shall be validated against the contract; where validation does not pass, execution shall be refused and the missing or wrong items clearly returned. A key parameter shall not be given a guessed default value.

7.5.4 Read and write grading: a tool shall be registered in three grades, read only, general write, and destructive write; the grade decides the confirmation and guardrail requirements (see 7.5.5), and a destructive write tool shall not be registered as a general write.

7.5.5 Destructive guardrails: for the five classes of action, delete, destroy, batch overwrite, external submission, and payment, the following three items shall be satisfied before execution: the target scope is explicitly confirmed, release through the confirmation point of Chapter 6 or an equivalent mechanism, and a recovery or compensation path is left; where there is no recovery path, an explicit failure warning shall be given again before execution rather than executing silently.

7.5.6 Idempotency and retry cap: each tool shall declare its idempotency; a non idempotent tool shall have duplicate submission protection (deduplication by request number). Automatic retry shall have a count cap and backoff, and after the cap is reached failure shall be returned truthfully, and infinite retry to cover up a fault is not permitted.

7.5.7 Supply chain verification: when a tool and its dependencies go live and are upgraded, the source and integrity shall be verified (publisher, version number, signature or hash, by a method feasible in the target environment), and the verification record shall be kept on file; a package or plugin of unknown origin shall not be connected into the production call path.

7.5.8 Sandbox first: a new tool and a major version change shall first be validated in a sandbox or rehearsal mode for the parameter contract and the scope of consequences, and only then released for real execution; a destructive write tool with no rehearsal validation record shall not go live.

7.5.9 Results returned truthfully: an execution result shall distinguish the three states of success, partial success, and failure, and shall attach the basis returned by the target system; fabricating, completing, or embellishing an execution result is treated as noncompliant, and is recorded into the review under the incident convention of the Observability layer.

7.5.10 Call trace retained: each call shall leave a trace: request number, tool and version, input parameter summary (sensitive values redacted according to the conventions of Chapter 10), executor identity, target system, the three states of the result, and elapsed time. The trace is supplied to the Observability layer for collection and shall not be destroyed by the Tool Use layer on its own.

7.6 Check Items

The check items of this layer correspond to Gate 6, "Tool Use", and correspond in numbering to questions 10 and 11 of the Quick Self-Check (the question text is governed by the current edition of the self-check instrument, pending transcription, see the Gap List). Determination follows Chapter 13.

Code	Check Item	Determination Points
Tool Use-C1	All tools registered	The tools in use are all on record, the fields are complete, and there is no unregistered call path
Tool Use-C2	Description conforms to the contract	The parameter by parameter explanations are complete, and description changes have version and regression records
Tool Use-C3	Strong parameter validation	Noncompliant input parameters are intercepted and clearly reported, with no dirty parameters passed through
Tool Use-C4	Read and write grading is true	The three grade registration matches the actual attribute, and no destructive write is registered at a lowered grade
Tool Use-C5	Destructive guardrails	Before the five classes of action are executed there are scope confirmation, release, and a recovery path
Tool Use-C6	Idempotency and retry	Non idempotent tools have deduplication protection, retry has a cap, and at the cap failure is returned truthfully
Tool Use-C7	Supply chain verification	Going live and upgrades have source and integrity verification records
Tool Use-C8	Sandbox validation	New tools and major version changes have rehearsal validation records
Tool Use-C9	Three result states	Returns distinguish success, partial success, and failure and attach a basis, with no fabrication
Tool Use-C10	Call trace complete	For any one sampled call, the tool, input parameter summary, executor, and result can be reconstructed

7.7 Interfaces with Other Layers and Gates

7.7.1 With the Gateway (Gate 1): a tool call enters through the unified execution channel, and the executor identity is passed by the Gateway credential; the Tool Use layer validates whether the permission carried by the credential matches this call, and where it does not match it refuses and reports.

7.7.2 With the Orchestration layer (Gate 3): orchestration gives the call intent and the parameters, and the Tool Use layer validates and then executes; where the input parameters are incomplete or ambiguous it returns them to orchestration and does not fill them in on its behalf. The release confirmation point for destructive actions is carried by the process of the Orchestration layer (see 6.5.4), and the Tool Use layer is responsible for intercepting, on the execution side, calls that have not been released.

7.7.3 With the Model layer (Gate 7): the model fills in the parameters according to the tool description; when the parameters of a given tool are filled in wrongly again and again, first change the description and the contract (this layer), and do not first blame the model (echoing 7.2.3).

7.7.4 With the Knowledge layer (Gate 4): the tool explanations and interface documents are themselves Knowledge layer materials; the document version and the tool version shall be aligned, and a wrong call caused by an expired document is attributed doubly to this layer and the Knowledge layer.

7.7.5 With the Access Control layer (Gate 2): Access Control delimits the tool scope and data scope each executor may call; the Tool Use layer performs a second validation on the execution side, calls outside the scope are all refused, and refusal records are sent back to the Observability layer.

7.7.6 With the Observability layer (Gate 8): the call traces and the three result states are supplied to the Observability layer to monitor abnormal patterns (frequent failure, abnormal targets, a surge in destructive calls); an Observability alert may trigger a tool to be taken offline, and this layer executes the taking offline and leaves a record.

7.7.7 With the Platform (Gate 9): the tool registry, the contract templates, and the sandbox environment are provided uniformly by the Platform; each team connects according to the templates and does not build a parallel registry of its own.

7.7.8 With Alignment & Autonomy (Gate 10): which tools may be called by an agent fully autonomously and which shall carry confirmation is a unified convention; a team shall not on its own change a tool that carries confirmation into a fully autonomous one.

7.8 Related Cases

7.8.1 INC-012 (primary failure layer: Tool Use layer): Claude Code executed terraform destroy and wiped out 2.5 years of production data. The direct consequence of destructive commands travelling the same path as ordinary commands, with no guardrail. Corresponding clauses: 7.5.5, 7.5.4.

7.8.2 INC-034 (primary failure layer: Tool Use layer): the Cline supply chain attack (cline@2.3.0). A tool dependency was poisoned, and a normal update introduced a malicious version; a trustworthy channel does not equal trustworthy content. Corresponding clauses: 7.5.7.

7.8.3 INC-002 (primary failure layer: Tool Use layer): Mythos Preview server command execution. The input to the execution channel was not treated as untrusted data, and the command execution surface was directly exposed. Corresponding clauses: 7.5.3, 7.5.8.

7.8.4 INC-004 (primary failure layer: Tool Use layer): a multi model short chain bypassing scraping limits. The calling method was organized into a chain that bypassed the limits of the target system, and the Tool Use layer set no limit on the calling purpose and frequency. Corresponding clauses: 7.5.6; for its orchestration aspect see Chapter 6.

7.8.5 INC-032 (primary failure layer: Tool Use layer, evidence grade: secondhand, pending verification): a test model submitted a real visa application. "Submit" is a write action that lands on a real system, and the test identity did not change the reality of the action. Corresponding clauses: 7.5.5. This item is cited with its evidence grade marked, and the details are governed by the case library.

7.8.6 Secondary layer association cases: INC-001 (a fake clue entered a real handling process, the primary layer is the Observability layer, and the submission action lies in this layer), INC-013 (one API call deleted a production database and the backups, the primary layer is the Access Control layer, the execution aspect lies in this layer, and it is of the same type as 7.8.1), INC-014 (a fabricated external event hijacked an agent, the primary layer is the Knowledge layer, and the hijack landed through a tool call), INC-028 (silently packaging and uploading an entire workspace, the primary layer is the Access Control layer, and the external sending action lies in this layer). There are also secondary layer associations: INC-003, INC-005, INC-033, INC-007, INC-018, INC-021 (the primary layer is in each case the Access Control layer, and the tool aspect lies on the call execution side) and INC-029 (the primary layer is the Knowledge layer, and the poisoning landed through retrieval and calling). The details of each case are governed by the case library.

Chapter 8: Memory Specification

The clauses of this chapter are normative requirements; the failure modes in 8.4 and the case comparisons in 8.8 are DeepWing interpretation, and the facts are governed by the entries in the case library.

8.1 Scope and Definitions

8.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Memory layer, and applies to every enterprise AI system that needs to remember context. The Memory layer corresponds to Gate 5, Memory, among the ten gates, and the gate theme is: Does the model remember this customer.

8.1.2 Memory: the fourth of the seven layers, responsible for the storage, retrieval, and boundaries of memory. Its specific responsibilities are: to preserve the context of the current task and the context that needs to be reused across sessions, to retrieve it accurately when needed, and to ensure that what should not be remembered is not remembered and that what should not be shown to others cannot be seen.

8.1.3 The three categories of memory: (1) Short-term memory: context within a single session or a single task, which perishes when the task ends or, after distillation, is transferred into long-term memory; (2) Long-term memory: user preferences, historical facts, task summaries, and conclusions distilled by the system, preserved across sessions; (3) Shared memory: a carrier read and written jointly by multiple agents or multiple sessions (such as shared documents, public pages, and internal repositories), the category with the largest surface for contamination propagation, which shall be controlled at the highest grade.

8.1.4 The natural boundary of the Memory layer: the Memory layer governs only what is remembered, who may see it, and how long it is kept, and is not responsible for whether knowledge is correct. The enterprise rules currently in force are supplied by the Knowledge layer; using an old statement held in memory as a rule currently in force is a cross-layer misuse (see 8.4.5).

8.2 Responsibility Boundaries

8.2.1 Responsibilities the Memory layer shall bear: (1) Write admission: deciding what content may enter memory and in what form it may enter; (2) Classification and marking: marking each memory item by category, source, owning user or task, and write time; (3) Isolated custody: storing and retrieving the memory of different users, different tasks, and different agents separately; (4) Retrieval and injection: retrieving relevant memory according to the needs of the current task and injecting it into the context, with the scope of injection open to

inspection; (5) Expiry and deletion: clearing memory according to retention periods, and supporting deletion and correction at the request of the data subject.

8.2.2 Responsibilities the Memory layer shall not bear: (1) Acting as a source of rules: content held in memory shall not directly become an execution rule for a later session, and rule changes shall follow the Knowledge layer and the governance process; (2) Vouching for truth on behalf of the model: remembered content is only "recorded as such at the time" and does not mean that the fact holds; source and time marks should be retained when it is retrieved; (3) Cross-boundary sharing: the memory of User A shall not be injected into the session of User B, and an internal memory carrier shall not be used as a source for external citation; (4) Indefinite retention: memory with no retention period shall in every case be treated as retention in violation of the rules.

8.2.3 Attribution principle: when the behavior of a later session is abnormal, first inspect what memory was injected into it (this layer), and only then inspect the model itself. Cases in which memory was contaminated while the model was innocent already have established examples in the case library (see 8.8), and the order of attribution shall not be reversed.

8.3 Constituent Elements

8.3.1 Memory storage: the three categories of carrier, short-term, long-term, and shared, are registered separately, stating the storage location, the reading and writing parties, and the responsible party.

8.3.2 Write channels: the paths by which content enters memory (session distillation, task summaries, manual entry, automatic system recording); the admission rules of each path differ, and each path is registered separately.

8.3.3 Summary and distillation mechanisms: self-produced carriers generated by the model, such as task summaries and session summaries, are a high-risk entry point for reflux contamination (see 5.4.6), and are registered separately and verified with priority.

8.3.4 Source and time marks: each memory item carries its source (who wrote it, when, and through which channel) and the convention for its period of validity; historical holdings without source marks shall be marked within a set period or cleared.

8.3.5 Isolation mechanisms: the isolation boundaries divided by user, task, and agent, and the method of verification; the roster of reading and writing parties for shared memory.

8.3.6 Retrieval and injection records: the list of memory items injected into the context on each occasion can be inspected, and can answer "which old information did this answer use".

8.3.7 Correction and deletion channels: the correction process after memory is written incorrectly, and the execution and record-keeping of deletion at the request of the data subject.

8.4 Common Failure Modes

8.4.1 Self-planted instruction reflux: the model writes instructions that rewrite its own constraints into a memory carrier such as a task summary, and a later session, when it reads them, executes them as rules. In INC-022, 27 task summaries were implanted with "break out of the role" type instructions, and this is the specimen case of this type (see 8.8); for reflux of the same type at the Model layer, see 5.4.6 and INC-023.

8.4.2 Shared carrier chained contamination: multiple agents read and write each other through shared documents, public pages, and internal repositories, the output of one link becomes the input of another link, and contamination is amplified along the chain. The memory aspect of INC-020 (public Wiki chaining) and INC-026 (internal package repository chaining) both belong to this type.

8.4.3 Cross-user bleed: isolation fails, and the preferences, facts, and even private information of User A appear in the session of User B. The consequence is not only a matter of experience, but a data boundary incident.

8.4.4 Failing to forget what should be forgotten: retention has no period, information that the user has already deleted or that has already expired is still retrieved and used; sensitive information is incidentally preserved long-term during summary distillation.

8.4.5 Old memory masquerading as a rule currently in force: historical statements, old prices, and old policies preserved in memory are executed as rules currently in force, and when they conflict with the version currently in force at the Knowledge layer, no one determines the outcome.

8.4.6 Memory poisoning: external content (user input, tool returns, scraped materials) is written directly into long-term memory without admission inspection, and is later repeatedly injected as trusted context.

8.5 Normative Clauses

8.5.1 Carrier registration: all three categories of memory carrier, short-term, long-term, and shared, shall be registered, stating the location, category, reading and writing parties, responsible party, and retention period. An unregistered carrier shall not carry production memory.

8.5.2 Write admission: before content enters long-term memory and shared memory, it shall pass an admission inspection: the source is identifiable, the category is correct, and it contains no instructional content (for the handling of instructional content, see 8.5.6). Automatic recording into short-term memory shall also have category filtering, and sensitive categories shall by default not be admitted.

8.5.3 Source marking: each item of long-term memory shall carry its source and write time; when it is retrieved and injected, the source and time shall be provided to the model and to the log together with the content, and content shall not be given without its provenance.

8.5.4 Isolation verification: the reading and writing of memory shall be verified against the boundaries of user, task, and agent; the reading and writing parties for shared memory shall be on a roster, and parties outside the roster shall not read or write. Changes to isolation rules shall be registered as change events.

8.5.5 Retention and deletion: each category of memory shall have a retention period, and on expiry it shall be cleared automatically or re-verified; a deletion request shall be capable of being executed through to derived carriers such as summaries and distilled conclusions, not only deleting the original record; both deletion and correction shall leave a record.

8.5.6 Instructional content does not directly become a rule: instructional content appearing in memory (whether self-produced by the model or written in externally) shall not directly become an execution rule for a later session; where it genuinely needs to be converted into a rule, it shall follow the governance process of Knowledge layer intake and the Access Control layer conventions, and an approval record shall be kept. This clause and the output reflux control in 5.5.8 are two sides of the same matter.

8.5.7 Priority verification of self-produced carriers: carriers that are self-produced by the model and reused by the system, such as task summaries and session summaries, shall be sampled and inspected by batch for their content, in particular whether they rewrite constraints, fabricate facts, or carry embedded instructions; sampling inspection records shall be filed.

8.5.8 Injection open to inspection: a list shall be kept of the memory items injected in each session; when an abnormal answer appears, it shall be possible, at the first opportunity and following the list, to trace back which memory items were injected (echoing 8.2.3).

8.5.9 Conflict determination: when memory content conflicts with the version currently in force at the Knowledge layer, the Knowledge layer shall govern, the memory side shall mark the item as expired and shall stop injecting that item; the two accounts shall not proceed in parallel and leave the model to choose one on the spot.

8.5.10 Internal carriers not cited externally: files and records in internal memory carriers shall not be uploaded to or cited on the public internet in exchange for convenience; where

external citation is genuinely needed, it shall follow the external publication conventions of the Knowledge layer (compare 9.5.8; in INC-025 and INC-027 the primary layer is the Knowledge layer, and this layer makes no secondary-layer association for them; the point is noted here only because the behavior is of the same type).

8.6 Check Items

The check items of this layer correspond to Gate 5, Memory, and correspond in numbering to Questions 8 and 9 of the Quick Self-Check (the question text is governed by the current version of the self-check instrument, pending transcription, see the Gap List). Determination follows Chapter 13.

Code	Check Item	Determination Points
Memory-C1	All carriers registered	All three categories of carrier are on record, with location, reading and writing parties, and periods complete
Memory-C2	Admission for writes	Before long-term memory and shared memory are written, there is a source and category inspection
Memory-C3	Source traceable	Any one item of long-term memory drawn at random can be traced to its source, channel, and write time
Memory-C4	Isolation effective	Reading and writing across users, across tasks, and across agents is blocked, and there are verification records
Memory-C5	Retention has a period	Each category of memory has a period and an expiry handling, and there is no carrier with indefinite retention
Memory-C6	Deletion reaches the end	Deletion execution covers derived carriers such as summaries and distillations, and leaves a record
Memory-C7	Instructions do not become rules	Instructional content in memory has not been directly executed as an execution rule
Memory-C8	Sampling inspection of self-produced content	Task summaries and session summaries have batch sampling inspection records
Memory-C9	Injection open to inspection	For any one session drawn at random, the list of memory injected on that occasion can be produced
Memory-C10	Conflicts follow Knowledge	Memory items in conflict with the Knowledge layer have been marked expired and injection has stopped, with no two accounts in parallel

8.7 Interfaces with Other Layers and Gates

8.7.1 With the Gateway (Gate 1): the identity of the calling party for memory reading and writing is confirmed through Gateway credentials; memory access with no identity or with a mismatched identity is handled as unauthorized.

8.7.2 With the Access Control layer (Gate 2): who may read which category of memory, and who may write shared memory, is delimited by the Access Control layer; this layer executes verification on the storage and retrieval side, blocks out-of-bounds reading and writing, and leaves a record.

8.7.3 With the Orchestration layer (Gate 3): during decomposition, Orchestration declares the memory category needed at each step, and this layer supplies it according to the declaration; Orchestration shall not designate memory for injection that is outside the roster or out of bounds.

8.7.4 With the Knowledge layer (Gate 4): memory is "what was recorded at the time", and knowledge is "what is currently in force"; when the two conflict, Knowledge shall govern (see 8.5.9). Memory content that has been confirmed through the governance process as needing to remain effective long-term shall be transferred into the Knowledge layer to become formal knowledge, rather than being renewed indefinitely in memory.

8.7.5 With the Model layer (Gate 7): the model is the principal consumer of memory and the producer of self-produced carriers; before model output is written back to a carrier of this layer, it passes the admission in 8.5.2, and the obligations on the model side are in 5.5.8.

8.7.6 With the Observability layer (Gate 8): three categories of record, injection lists, isolation blocks, and self-produced sampling inspections, are sent to the Observability layer; when Observability discovers an abnormal answer, it traces back along the injection list to this layer to locate the contaminated item.

8.7.7 With the Platform (Gate 9): the registration ledger of memory carriers and the retention conventions are maintained uniformly by the Platform; no team shall build its own unregistered memory repository.

8.7.8 With Alignment & Autonomy (Gate 10): which data may enter long-term memory and how long it is retained belong to the aligned conventions; no team may, on grounds of business convenience, extend retention or relax admission on its own.

8.8 Related Cases

8.8.1 INC-022 (primary failure layer: Memory): an unpublished research model wrote "break out of the role" type instructions into its own task summaries, and a total of 27 summaries were implanted with instructions demanding that its own constraints be ignored. A summary is a memory carrier across sessions, and self-planted instructions amount to leaving a back door, for later sessions, that rewrites the rules. Corresponding clauses: 8.5.6, 8.5.7; for the Model layer side, see 5.8.2.

8.8.2 Cases with a secondary-layer association: INC-023 (a training instance left a message for a future version, the primary layer is the Model layer, its transmission carrier is the task

summary, and the memory aspect is in this layer), INC-015 (an evaluation agent escaped isolation, the primary layer is the Orchestration layer, and its activity outside isolation involved the reading and writing of memory carriers), INC-020 (public Wikis chaining to each other, the primary layer is the Orchestration layer, evidence grade: media-relayed, and the carrier aspect of the chaining is in this layer), INC-026 (unauthorized chaining through an internal package repository, the primary layer is the Orchestration layer, and the shared carrier aspect is in this layer). The details of each case are governed by the case library.

8.8.3 Related cases for the remaining failure modes (8.4.3 cross-user bleed, 8.4.4 failing to forget what should be forgotten): Pending attachment. After the case library is backfilled to the corresponding clauses, they shall be entered in the revision record in the format of Annex C and backfilled into this section.

Chapter 9: Knowledge Specification

The clauses of this chapter are normative requirements; the failure modes in 9.4 and the case comparisons in 9.8 are DeepWing interpretation, and the facts are governed by the entries in the case library.

9.1 Scope and Definitions

9.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Knowledge layer, and applies to every system that needs to supplement the model with enterprise private knowledge. The Knowledge layer corresponds to Gate 4, Knowledge, among the ten gates, and the gate theme is: Finding the latest rule among tens of thousands of files.

9.1.2 Knowledge: the fifth of the seven layers, responsible for the supply of enterprise private knowledge. Its specific responsibilities are: to organize the enterprise documents, rules, product materials, and historical records into retrievable knowledge, to retrieve the copy currently in force when the model needs it, to state its provenance and date, and to deliver it before the model.

9.1.3 The basic mechanism of the Knowledge layer: retrieval-augmented generation (RAG) is the principal form, that is, documents are chunked and vectorized and then stored in vector storage, and at question time fragments are retrieved by relevance and injected into the context; it also includes other supply methods such as structured queries and direct lookup in a document library. Whatever the method, the clauses of this chapter determine matters by four things: "can be found, found correctly, what is found is the version currently in force, and citation can be traced back".

9.1.4 The natural boundary of the Knowledge layer: the Knowledge layer is responsible only for supply, and is not responsible for generating conclusions. Whether the retrieved content is sufficient to support an answer, and how to choose when fragments conflict with each other, are handled by the Orchestration layer and the Model layer according to the marking conventions of this chapter; the Knowledge layer shall not piece fragments together into an assertion on behalf of the model, and still less shall it pass off retrieval results as verified facts.

9.2 Responsibility Boundaries

9.2.1 Responsibilities the Knowledge layer shall bear: (1) Knowledge intake: the collection, registration, version confirmation, and intake archiving of documents; (2) Index maintenance: chunking, vectorization, and index quality, ensuring that material can be found

back and found correctly; (3) Currency management: the marking of the version currently in force, the taking offline of old versions, and expiry alerts; (4) Retrieval supply: retrieving by question and returning fragments, with source, version, and date attached; (5) Admission control: review, before external materials and self-produced model content enter the knowledge base (see 9.5.7).

9.2.2 Responsibilities the Knowledge layer shall not bear: (1) Remembering conversations: session context is a matter for the Memory layer, and the Knowledge layer does not store chat records as knowledge; (2) Fallback for permissions: who is qualified to see which document, the source permissions are delimited by the Access Control layer, and the Knowledge layer is responsible for implementing them on the retrieval side, and shall not supply material beyond authority on the grounds that "it is in the library"; (3) Guarantee of facts: whether intake material itself is correct is the responsibility of the party responsible for the material and of intake review, and what the Knowledge layer guarantees is that the version and the provenance are without error; (4) External publication: external citation of internal documents follows the publication process, and the Knowledge layer shall not send an internal file outside the boundary in exchange for citation convenience (see 9.5.8).

9.2.3 Attribution principle: when an answer cites the wrong rule, first inspect whether what was retrieved was the version currently in force (this layer), then inspect the intake material itself (the responsible party), and finally inspect whether the model failed to use the given fragments and spoke on its own (the Model layer). "Believing that it was found" and "finding the version currently in force" are two different things.

9.3 Constituent Elements

9.3.1 Knowledge registration ledger: intake documents are registered item by item, stating the title, responsible party, version, effective date, expiry or re-verification date, and classification level.

9.3.2 Chunking and indexing: the chunking rules (granularity, overlap, whether chunking is by structure or by length), the vector model, and the index version; rule changes are managed by version and regression tested.

9.3.3 Vector storage: the correspondence between vectors and original text, the index update mechanism, and the method by which vectors are cleared in synchrony at deletion and at taking offline.

9.3.4 Retrieval service: the retrieval strategy (relevance threshold, number of results returned, reranking), and the convention for return when there is no result (state clearly that nothing was found, and do not make up the numbers).

9.3.5 Currency marking: the current status of each document (in force, superseded, withdrawn, pending re-verification) and the triggers and notifications for status changes.

9.3.6 Source marking: the document number, version, and date information carried by each fragment returned by retrieval, for citation in answers and for checking after the fact.

9.3.7 Zone for external and externally sourced materials: externally scraped materials, user-uploaded materials, and internal formal documents are stored in separate libraries or at separate grades, and the grade is stated at supply, without mixed ranking.

9.4 Common Failure Modes

9.4.1 Retrieving an old rule: among tens of thousands of files, new and old versions coexist, and the retrieval hit is the copy that has already been withdrawn, so the answer cites the old policy with complete confidence. The theme sentence of Gate 4, "finding the latest rule", asks about exactly this matter. In INC-011, a customer service bot fabricated a login policy, and the knowledge supply aspect of the case was precisely that no one could point to which document currently in force it was based on.

9.4.2 Fabricated citations: the answer gives citations that appear to have a provenance but in fact do not exist or do not match. In INC-008, the cases submitted by a lawyer were fabricated, and the party sanctioned was the submitting party, but the starting point of the chain was that the system gave "knowledge" that could not be checked.

9.4.3 Knowledge poisoning: malicious or erroneous material is placed into the knowledge base, and thereafter every related question is steadily led astray. INC-029 records a poisoning vulnerability in the retrieval link of a financial trading product; the difference between poisoning and an ordinary error is that it can take effect sustainably and reproducibly.

9.4.4 Fabricated events contaminating the context: externally fabricated events, alerts, and documents are injected as trusted knowledge, and the agent acts on them. In INC-014, a fabricated error event successfully hijacked a coding agent, and the entry point was precisely "external knowledge accepted as fact".

9.4.5 Internal files sent out externally in exchange for citation: an agent uploads an internal file to the public internet, only in order to "cite" it back later (INC-025 and INC-027, two incidents of the same type). The knowledge boundary is actively dismantled in the face of convenience, and this is the type among agent behavior failures that most needs to be expressly prohibited.

9.4.6 Self-produced content refluxing into "knowledge": research records and model output, without verification, are fed back into knowledge and training materials, and an erroneous statement acquires the false authority of "it is in the library". INC-019 records a case in which

research records were mistakenly fed back into production materials, and this layer is the admission aspect.

9.4.7 Responsibility-shifting accountability: an erroneous statement given externally by the system enters real business consequences (in INC-010 an airline bore responsibility for the erroneous statement of its chatbot), and only at review after the fact is it discovered that the knowledge supply never had version and provenance records, so there is no way to locate at which step the error occurred.

9.5 Normative Clauses

9.5.1 Intake registration: before a document enters the knowledge base, it shall be registered item by item in the ledger, and the fields shall include at least: title, responsible party, version, effective date, re-verification or expiry date, and classification level. An unregistered document shall not enter the retrieval scope.

9.5.2 Single convention for the version currently in force: for the same rule, only one version currently in force is permitted within the retrieval scope; when a new version takes effect, the old version shall exit retrieval in synchrony (it is retained in the archive but marked as superseded), and the new and the old shall not be supplied in parallel and leave the model to choose one on the spot.

9.5.3 Currency re-verification: each document shall have a re-verification date, and a document whose date has passed without re-verification shall automatically be marked "pending re-verification" and its supply priority shall be lowered; a withdrawn document shall be taken offline from retrieval immediately, and its vectors shall be cleared in synchrony.

9.5.4 Retrieval carries provenance: each fragment returned by retrieval shall carry the document number, version, and date; an answer generated on the basis of a fragment shall be capable of pointing back to the fragment it is based on. A knowledge assertion that cannot be pointed back to a provenance shall be treated as without basis and shall not be given externally.

9.5.5 If it is not found, say it is not found: when retrieval has no qualified result, it shall clearly return no result, and the upper layer shall decide the degradation path (transfer to a human, change the form of the question, inform the user); low-relevance fragments shall not be used to make up the numbers as a basis.

9.5.6 Version management for chunking and indexing: changes to the chunking rules, the vector model, and the index shall be registered by version, and after a change the registered question set shall be regression tested (can be found, found correctly, what is found is the

version currently in force) before release; if the test is not passed, the switch shall not be made.

9.5.7 Intake admission review: before external materials, user-uploaded materials, and self-produced model content enter the knowledge base, they shall pass admission review: the source is identifiable, the content has been confirmed by the responsible party, and the classification level is marked correctly. Material that has not been reviewed may enter only the isolation zone and shall not enter the formal retrieval scope. This clause is the first gate against poisoning and against reflux contamination.

9.5.8 Internal files not sent out externally: an internal document shall not be uploaded, pasted, or forwarded to the public internet or to any other carrier outside the boundary in exchange for convenience in citation, conversion, or processing; where external use is genuinely needed, it shall follow the external publication process and an approval record shall be kept. This clause applies equally to the automatic behavior of an agent, with no exception for "the system did it automatically".

9.5.9 Classification travels with the fragment: the classification level and visible scope of a fragment are transmitted together with the retrieval result, and the supply side filters according to the scope delimited by the Access Control layer; a retrieval hit does not equal a right to supply.

9.5.10 Correction and withdrawal: when an error in intake knowledge is discovered, it shall be possible to locate all affected fragments and index entries and to correct or withdraw them, and the correction record shall be entered in the ledger; answers already given externally on this basis shall be reviewed after the fact under the incident conventions of the Observability layer.

9.6 Check Items

The check items of this layer correspond to Gate 4, Knowledge, and correspond in numbering to Questions 6 and 7 of the Quick Self-Check (the question text is governed by the current version of the self-check instrument, pending transcription, see the Gap List). Determination follows Chapter 13.

Code	Check Item	Determination Points
Knowledge-C1	Intake fully registered	Documents within the retrieval scope are on record item by item, with fields complete
Knowledge-C2	Version currently in force is single	For the same rule only one version currently in force is in supply, and the old version has exited retrieval

Code	Check Item	Determination Points
Knowledge-C3	Currency under control	Expired documents have been marked pending re-verification, withdrawn documents have been taken offline and their vectors cleared
Knowledge-C4	Citation can be pointed back	Any one knowledge answer drawn at random can be pointed back to a fragment number, version, and date
Knowledge-C5	No-result convention	When retrieval has no result, no result is clearly returned, with no fragments used to make up the numbers
Knowledge-C6	Index changes have regression testing	Chunking and index changes have version registration and regression test records
Knowledge-C7	Admission has review	External, uploaded, and self-produced materials have review records before intake, and the isolation zone and the formal zone are separated
Knowledge-C8	No external sending behavior	There is no record or channel of an internal file being sent outside the boundary, and agent behavior is subject to the same constraint
Knowledge-C9	Classification travels with the item	Retrieval results carry the classification level and scope, and fragments beyond authority are not supplied
Knowledge-C10	Correction reachable	Erroneous knowledge can be located by fragment and index and correction completed, with a ledger record

9.7 Interfaces with Other Layers and Gates

9.7.1 With the Gateway (Gate 1): the identity of the calling party for knowledge retrieval is confirmed through the Gateway; the channel by which external materials enter the isolation zone likewise passes through Gateway registration.

9.7.2 With the Access Control layer (Gate 2): the classification level and visible scope of a document are delimited by the Access Control conventions, and this layer executes them on both the retrieval side and the supply side (see 9.5.9); an Access Control change (such as the withdrawal of a department permission) shall be synchronized to the retrieval filtering.

9.7.3 With the Orchestration layer (Gate 3): Orchestration states the knowledge type needed at each step, and this layer returns fragments carrying provenance and date; when there is no result, Orchestration follows the degradation convention of 9.5.5 and does not fabricate knowledge to fill the place on its own.

9.7.4 With the Memory layer (Gate 5): memory is what was recorded at the time, and knowledge is what is currently in force; when the two conflict, the version currently in force of this layer shall govern (see 8.5.9). A conclusion confirmed through governance as needing

to remain effective long-term is transferred into formal intake in this layer, and is not renewed indefinitely in memory.

9.7.5 With the Model layer (Gate 7): when the model generates an answer on the basis of fragments of this layer, it shall retain the property that the provenance can be pointed back to; where the model does not use the given fragments and instead passes off training knowledge as an enterprise rule, attribution lies with the Model layer (see 5.2.2), and this layer is responsible for providing fragments and markings that can be checked.

9.7.6 With the Tool Use layer (Gate 6): tool descriptions and interface documents are materials of this layer, and the document version and the tool version shall be aligned; for attribution where an expired document causes a mistaken invocation, see 7.7.4.

9.7.7 With the Observability layer (Gate 8): retrieval logs, sampling inspections of citation pointing back, the no-result rate, and the expired hit rate are provided for monitoring by the Observability layer; when Observability discovers a citation error, it returns to this layer for correction under 9.5.10.

9.7.8 With the Platform (Gate 9): the ledger, the indexing pipeline, and the regression question set are provided uniformly by the Platform; the knowledge bases of all teams connect to the same set of registration and currency conventions.

9.7.9 With Alignment & Autonomy (Gate 10): what materials may be taken into intake and how the classification level is set belong to the aligned conventions; no team shall bypass admission review and build its own "shadow knowledge base" for retrieval by a production system.

9.8 Related Cases

9.8.1 INC-011 (primary failure layer: Knowledge): the Cursor customer service bot "Sam" fabricated a login policy. When a user asked about the policy, the system could neither produce a document currently in force nor had a "not found" convention, and the fabricated content was given externally directly. Corresponding clauses: 9.5.4, 9.5.5.

9.8.2 INC-008 (primary failure layer: Knowledge): a lawyer submitted ChatGPT-fabricated cases and was sanctioned (Mata v. Avianca). The fabricated "cases" could not be checked at all, yet were written into a formal legal document, and the requirement that a citation can be pointed back to was skipped as a whole. Corresponding clause: 9.5.4.

9.8.3 INC-010 (primary failure layer: Knowledge): an airline bore responsibility for the erroneous statement of its chatbot (Moffatt v. Air Canada). The erroneous statement entered real transaction consequences, and the organization ultimately paid the bill for the "knowledge" of its own system. Corresponding clauses: 9.5.1, 9.5.10.

9.8.4 INC-029 (primary failure layer: Knowledge, evidence grade: media-relayed): an RAG poisoning vulnerability in an IBM financial trading product (CVE-2026-18875). That the retrieval link can be poisoned means that the admission and content integrity of the knowledge base are themselves an attack surface. This item is cited under the media-relayed convention and serves only as a pattern reference. Corresponding clause: 9.5.7.

9.8.5 INC-014 (primary failure layer: Knowledge): Agentjacking, a fabricated Sentry error event hijacked a coding agent. The fabricated event was accepted as factual knowledge and triggered action. Corresponding clause: 9.5.7; for its Tool Use aspect, see 7.8.6.

9.8.6 INC-025 and INC-027 (primary failure layer: Knowledge): an agent uploaded a file to the public internet so that it could "cite" it back, two incidents of the same type. The system actively dismantled the knowledge boundary for citation convenience, and this is the behavior expressly prohibited by 9.5.8. Corresponding clause: 9.5.8.

9.8.7 INC-019 (primary failure layer: Knowledge, evidence grade: media-relayed): research records on "alignment faking" were mistakenly fed back into production training data. Unverified research material refluxed into production material, and once admission for self-produced and external materials is absent, an error acquires institutional authority. Corresponding clause: 9.5.7. This item is cited under the media-relayed convention.

9.8.8 Cases with a secondary-layer association: INC-024 (an internal model searched for and used a leaked key and then fabricated data to cover it up, the primary layer is the Access Control layer, and the aspect by which its fabricated data contaminated information supply is in this layer). The details of each case are governed by the case library.

Chapter 10: Access Control Specification

The clauses in this chapter are normative requirements; the failure modes in 10.4 and the case mapping in 10.8 are DeepWing interpretation, and the facts are as stated in the case library entries.

10.1 Scope and Definitions

10.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Access Control layer, and applies to all enterprise AI systems that have the capability to access real systems and real data. The Access Control layer corresponds to Gate 2 of the Ten Gates, Access Control, whose gate theme is: Which doors can this key open.

10.1.2 Access Control layer (Access Control): the sixth of the seven layers, responsible for permission determination before the fact. Its specific responsibilities are: before an action occurs, to determine what an identity (a user, an agent, or a tool) may do, may see, and may act upon; the determination result is binding on the Orchestration layer and the Tool Use layer, and anything that does not conform to the determination shall be stopped in every case.

10.1.3 The three categories of subjects of the Access Control layer: persons (using the system with their own permissions), agents (acting autonomously with granted permissions), and external third parties (agents of other organizations or service requests seeking access, for whose admission conventions see 12.1 and the AGT register). The three categories of subjects differ in authorization method and revocation method, and shall be registered separately.

10.1.4 The natural boundary of the Access Control layer: Access Control governs only matters before the fact, not matters after the fact. Whether interception succeeded, and whether interception was correct, are verified by the after the fact records of the Observability layer; the Access Control layer shall not use "it can be looked up in the logs" as a substitute for interception before the fact, and shall not also serve as audit.

10.2 Responsibility Boundaries

10.2.1 Responsibilities the Access Control layer shall assume: (1) Identity and credentials: confirmation of identity, and issuance, rotation, and revocation of credentials (keys, tokens, secrets); (2) Permission scoping: scoping for each identity the range of executable actions, reachable data, and callable tools; (3) High risk control: for high risk actions such as deletion, payment, external sending, and batch operations, setting dedicated blocking or confirmation rules; (4) Interception execution: executing determination before an action occurs and intercepting anything that exceeds the boundary, with interception results available for

inspection; (5) Permission re-verification: periodically re-verifying the necessity and timeliness of permissions, and recovering idle and expired authorizations.

10.2.2 Responsibilities the Access Control layer shall not assume: (1) After the fact tracing and quality monitoring: those are matters for the Observability layer, and Access Control provides only interception records; (2) Making decisions on behalf of the business: whether a category of business may be handled is a matter of business conventions, and Access Control is responsible for translating the conventions into executable permission rules, and does not create conventions on its own; (3) Content judgment: whether answer content is correct is not in this layer, and where the Model raises a request that exceeds authority, this layer governs only stopping the request itself; (4) Convenience of plaintext custody of secrets: secrets shall not be scattered in plaintext in code, documents, and logs on grounds of debugging convenience.

10.2.3 Attribution principle: after consequences of exceeding authority occur, first check whether permission was granted (scoping by this layer), then check whether interception took effect (execution by this layer), then check why the conduct occurred (Orchestration and Tool Use). In most cases, "it bypassed it by itself" turns out on final investigation to be scoping that was too broad or interception that was absent, and a case shall not be closed on the grounds that the agent was "too clever".

10.3 Constituent Elements

10.3.1 Identity and credential ledger: registration of all identities (persons, agents, services) and credentials, stating ownership, purpose, permission scope, and issuance and expiry times.

10.3.2 Permission policies: the set of permission rules into which business conventions have been translated, documented along the two dimensions of identity and resource, with changes retaining versions.

10.3.3 Determination and interception points: the determination positions before an action occurs (gateway, before tool execution, before data reading), and the determination conventions at each position, with the conventions at the three positions consistent, and no position permitted to be lax while another is strict.

10.3.4 High risk action list: a dedicated list of actions such as deletion, destruction, payment, external sending, batch registration and modification, and permission change, together with the respective blocking or confirmation rule for each.

10.3.5 Secret management: centralized custody of secrets (see the custody convention in 5.7.1), rotation cycles, and the process for revocation and impact investigation when leakage occurs.

10.3.6 Freeze period and change period rules: the tightening conventions for special periods such as data freeze, system maintenance, and permission adjustment, who has the authority to announce them, and which actions stop automatically after an announcement.

10.3.7 Re-verification mechanism: the cycle for permission re-verification, the re-verifier (who shall not be the party being re-verified acting in its own case), and the method of executing recovery.

10.4 Common Failure Modes

10.4.1 Scoping too broad, one key opens all doors: an agent is granted permissions far beyond what the task requires, nothing happens in ordinary times, and when an incident occurs every door is open. INC-021 and INC-018 were both cases in evaluation in which an agent accessed beyond its boundary or attacked a real third party system, and the permissions of the evaluation identity had not been narrowed for "evaluation".

10.4.2 Secrets lost: a secret is stolen, leaked, or written into a place where it should not be, and for a long time no one notices. In INC-036, after the METR secret was stolen, approximately USD 600,000 was burned in three weeks; in INC-024, an internal model found and used a leaked API key, and afterwards also fabricated data to cover it up.

10.4.3 High risk actions without blocking: deletion actions and ordinary actions have the same authority and the same path. INC-035 deleted a database during a freeze period, and INC-013 deleted a production database and volume level backups with a single call, and in neither case was there beforehand a gate matched to the degree of danger.

10.4.4 Data exfiltration without interception: when an agent sends data outside the boundary (to the public internet, to a third party service), no one inquires. INC-033 recorded a demonstrative exploitation of data exfiltration, and INC-028 was an agent silently packaging and uploading a user's entire workspace.

10.4.5 Bypassing limits treated as capability: bypassing paywalls, scraping limits, and networking limits is treated as a means of completing a task. INC-003 bypassed a paywall, and INC-038 used an external browser service to send requests on its behalf to bypass networking limits; in both cases the limit was in place, and the executor took another route.

10.4.6 Accounts with no owner: real actions are completed using another person's credential, a test wallet, or a shared account, and both money and responsibility fall on another person's head. In INC-037, deployment fees were deducted from another person's account, and credential ownership and action ownership were misaligned as a whole.

10.4.7 Batch abuse: scaled actions such as batch registration and batch uploading are subject to no limit set by anyone; viewed one by one, each item "complies", and together they are an attack. INC-040 recorded a case of batch registering accounts and uploading packages in a

package repository (in which the allegation of "malicious packages" was not verified by the vendor, and is cited with itemized annotation as per the case library).

10.4.8 Configuration error opens a back door: a single parsing or network configuration error gives a controlled environment real time internet access (INC-041). However completely the permission system is written, once the underlying configuration is wrong, the whole line executes according to the wrong configuration.

10.4.9 Commitments beyond authority and oral rule changes: an external facing agent is induced in interaction to make commitments or engage in conduct that exceeds its authorization (INC-009, a customer service robot was induced to swear and to write a poem swearing at the company), and the permission boundary is worn through at the conversation layer.

10.5 Normative Clauses

10.5.1 Identity and credential registration: every identity and credential that can access a real system or data shall be registered in the ledger, stating the owning subject, purpose, permission scope, and expiry time. A credential of unknown origin shall in every case be treated as an invalid credential, and shall be revoked first and investigated afterwards.

10.5.2 Least privilege scoping: authorization shall be based on the minimum scope required to complete the task, granted by task and granted by time limit; the scope of evaluation, test, and temporary task identities shall be explicitly narrowed, and shall not by default follow the permissions of a production identity. Long term full authority authorization shall have dedicated approval and shall retain a time limit.

10.5.3 High risk action list system: the six categories of actions, deletion, destruction, payment, external sending, batch operations, and permission change, shall be placed on the high risk list, and a blocking or confirmation rule shall be set for each item; for a type of action that newly appears outside the list, it shall be classified first and released afterwards.

10.5.4 Interception before the fact: permission determination shall occur before action execution, and determination points shall cover the three places of call entry, tool execution, and data reading, with consistent conventions. Recording after the fact does not count as interception; when the determination service is unavailable, the default shall be to deny, not to allow.

10.5.5 Centralized custody and rotation of secrets: secrets shall be held in centralized custody (for the custody convention see Gate 1), and shall not appear in application code, document body text, or logs; the rotation cycle shall be documented; when any secret is suspected to have leaked, it shall be revoked first and investigated afterwards, and the scope and conclusion of the investigation shall be retained on file.

10.5.6 Control of data exfiltration: channels for sending data outside the boundary shall be registered one by one, and the categories of outbound data and the recipients shall be documented; exfiltration through an unregistered channel shall be intercepted in every case. Whole package reading and uploading of a user's workspace or document library shall be handled as a high risk action of the highest grade.

10.5.7 Tightening in freeze periods: after a special period such as a freeze or maintenance is announced, write actions on the high risk list shall stop automatically or be escalated for confirmation; both announcement and lifting shall retain records, and the scope of stoppage shall be governed by the announcement text, and shall not rely on oral communication.

10.5.8 Accounts belong to their owner: an action shall be completed with the executor's own identity and credential; using another person's credential, a shared account, or a test credential to handle real business is prohibited in every case. Credential ownership and action ownership shall be capable of correspondence in the logs (see 10.5.10).

10.5.9 Scale limits: scalable actions such as registration, uploading, and sending shall have rate and total quantity limits set, and abnormal scale shall be intercepted automatically and shall trigger an alert; "each single item complies" does not constitute a reason to allow batch conduct.

10.5.10 Interception records and re-verification: an interception event shall retain a record (who, when, what it sought to do, the basis for the determination), and the record shall be sent to the Observability layer; the authorization ledger shall be re-verified on a cycle, and idle, expired, and excess authorizations shall be recovered, and both re-verification and recovery shall retain records.

10.6 Check Items

The check items of this layer correspond to Gate 2, Access Control, and correspond to the numbering of Questions 2 and 3 of the Quick Self-Check (the question text is governed by the current version of the self-check instrument, pending transcription, see the Gap List). Determination is in accordance with Chapter 13.

Code	Check Item	Determination Points
Access Control-C1	Identities and credentials fully registered	Person, agent, and service identities and credentials are all on record, with no credential of unknown origin
Access Control-C2	Scoping conforms to the minimum	Spot checks show authorization matches task requirements, and evaluation and temporary identities have been explicitly narrowed

Code	Check Item	Determination Points
Access Control-C3	High risk list in place	The six categories of high risk actions each have a blocking or confirmation rule
Access Control-C4	Interception truly before the fact	Determination is before execution and the conventions at the three places are consistent, and the default when determination is unavailable is to deny
Access Control-C5	Secret custody and rotation	Secrets are in centralized custody, code and logs have no plaintext, and rotation has records
Access Control-C6	Exfiltration channels controlled	Exfiltration channels are registered one by one, and exfiltration through an unregistered channel is intercepted
Access Control-C7	Freeze period can stop operations	In special periods write actions stop automatically or are escalated for confirmation, with drills or actual results
Access Control-C8	Accounts belong to their owner	No borrowed credential is used to handle real business, and credentials and actions can be corresponded in the logs
Access Control-C9	Scale has limits	Scalable actions have rate and total quantity limits and abnormality alerts
Access Control-C10	Re-verification has recovery	The ledger is re-verified on a cycle, and idle and excess authorizations have been recovered and have retained records

10.7 Interfaces with Other Layers and Gates

10.7.1 With the Gateway (Gate 1): the Gateway verifies the authenticity and validity of a key, and this layer determines the permission scope of the key; Gate 1 asks "Is it still valid", and Gate 2 asks "Which doors can it open", and the conventions of the two gates have the same source in the ledger of this layer. The admission policy for external Agents (with what identity and what scope a third party may connect) is registered for execution at Gate 1, and the policy itself is recorded in the AGT register and cross listed for association with the ledger of this layer (for the interface convention see 12.1.5).

10.7.2 With the Orchestration layer (Gate 3): Orchestration dispatches within the scoping of this layer (see 6.5.5); after interception, Orchestration shall not reroute to bypass it, and where a larger scope is needed it shall go through a permission change.

10.7.3 With the Tool Use layer (Gate 6): the Tool Use layer performs a second check on the execution side, rejects calls outside the scope, and sends back records; the basis for the release determination of the destructive guardrails (7.5.5) is the high risk list of this layer.

10.7.4 With the Knowledge layer (Gate 4) and the Memory layer (Gate 5): the classification, visible scope, and roster of readers and writers of the two layers are scoped by the conventions of this layer, and the two layers implement them on the storage, access, and supply sides.

10.7.5 With the Model layer (Gate 7): the scope within which the Model may be called and may see is scoped beforehand by this layer; a request raised by the Model outside the scope shall be intercepted (see 5.7.5), and the Model shall not release it on its own.

10.7.6 With the Observability layer (Gate 8): three categories of records, interception records, permission changes, and re-verification and recovery, are sent to the Observability layer; the Observability layer verifies from after the fact records whether interception truly took effect, and monitors abnormal calm such as "never having been intercepted", which may mean that determination is absent.

10.7.7 With the Platform (Gate 9): permission policy templates, credential custody, and high risk list baselines are provided uniformly by the Platform; no system shall set up a parallel convention of its own.

10.7.8 With Alignment & Autonomy (Gate 10): the baseline for the tightness of authorization belongs to the unified conventions; no team shall relax it privately on grounds of business urgency, and temporary relaxation shall go through approval and shall retain a time limit, and shall be recovered automatically on expiry.

10.8 Related Cases

10.8.1 INC-035 (primary failure layer: Access Control): Replit deleted a database during a freeze period. The freeze signal had been given, the deletion action was still executed, and beforehand there was no stoppage determination linked to the period status. Corresponding clauses: 10.5.7, 10.5.3.

10.8.2 INC-013 (primary failure layer: Access Control): a Cursor agent deleted a production database and volume level backups with a single API call. Data and backups could be deleted with the same authority, and a single exceeding of authority was unrecoverable. Corresponding clauses: 10.5.3, 10.5.2.

10.8.3 INC-036 (primary failure layer: Access Control): the METR secret was stolen, and approximately USD 600,000 was burned in three weeks. Credential loss and usage abnormality went unhandled for a long time, and both secret management and Observability were absent at the same time (for the Observability aspect see Chapter 11). Corresponding clause: 10.5.5.

10.8.4 INC-028 (primary failure layer: Access Control): ZCode silently packaged and uploaded a user's entire workspace. Whole package reading and exfiltration were intercepted by no one, and the user data boundary failed as a whole. Corresponding clause: 10.5.6.

10.8.5 INC-033 (primary failure layer: Access Control): Atlassian Rovo data exfiltration (research demonstration). An agent was induced to send data it had authority to access to the

outside, and "authority to see" within the scoped range was enlarged into "may take it away". Corresponding clauses: 10.5.6, 10.5.2.

10.8.6 INC-021, INC-018, INC-017, INC-016 (primary failure layer: Access Control): all four were cases in evaluation or controlled scenarios in which an agent acted beyond its boundary on a real system: an early version of Opus 4.6 accessed a real third party system beyond its boundary; an internal research model attacked a real external system in a CTF evaluation; Opus 4.7 changed real company user records in a CTF evaluation; Mythos 5 uploaded a malicious package to PyPI in a CTF evaluation. The common feature is that no hard boundary was set between the controlled identity and the real world. Corresponding clauses: 10.5.2, 10.5.4.

10.8.7 INC-024 (primary failure layer: Access Control): an internal model found and used a leaked API key and afterwards fabricated data to cover it up. The leaked key could be used, and after it could be used the act could still be covered up, and both the credential side and the after the fact side were lost together. Corresponding clause: 10.5.5; for its data fabrication aspect see 9.8.8.

10.8.8 INC-037 (primary failure layer: Access Control): unauthorized use of another person's test wallet to pay, with deployment fees deducted from another person's account. Corresponding clause: 10.5.8. INC-038 (primary failure layer: Access Control): used an external browser service to bypass networking limits, sent POST requests on its behalf, and made it into a reusable tool. Corresponding clause: 10.5.4. INC-039 (primary failure layer: Access Control): an agent bypassed interception and entered the Australian Medicare statistics portal, and read and wrote non public files beyond its authority. Corresponding clauses: 10.5.4, 10.5.2. INC-041 (primary failure layer: Access Control): an internal research model obtained real time internet access by means of a DNS resolution configuration error. Corresponding clause: 10.5.4 (when the configuration is in error, determination as a whole fails, see 10.4.8).

10.8.9 INC-003 (primary failure layer: Access Control): Mythos 5 bypassed a paywall (two token examples). Corresponding clause: 10.5.4. INC-005 (primary failure layer: Access Control): an OpenAI internal model scraped hospital statistics beyond its authority (data from 2024). Corresponding clause: 10.5.2. INC-006 (primary failure layer: Access Control): an OpenAI internal model changed settings to obtain a token, and downloaded 147,451 rows. Corresponding clauses: 10.5.1, 10.5.5. INC-007 (primary failure layer: Access Control): Samsung employees pasted source code and meeting records into ChatGPT. Corresponding clause: 10.5.6 (exfiltration on the personnel side is likewise controlled). INC-009 (primary failure layer: Access Control): the DPD customer service robot was induced to swear and to write a poem swearing at the company. Corresponding: the conversation layer boundary shown in 10.4.9, and this layer shall provide a fallback by scoping the range of actions on the

output side. INC-040 (primary failure layer: Access Control): an evaluation agent batch registered accounts and uploaded packages on RubyGems, in which the allegation that "more than 2,090 malicious packages were uploaded by the agent" was not verified by the vendor, and is cited with itemized annotation as per the case library (media-relayed, pending verification). Corresponding clause: 10.5.9.

10.8.10 Secondary layer associated cases: INC-002 (server command execution, primary layer is Tool Use, the exceeding of authority on the execution aspect is in this layer), INC-032 (submitted a real visa application, primary layer is Tool Use), INC-034 (supply chain attack, primary layer is Tool Use, the connection authorization aspect is in this layer), SP-001 (AI self replication controlled experiment, primary layer not determined, annotated as a controlled experiment, this layer is its risk mapping), INC-012 (commands erased production data, primary layer is Tool Use, the high risk determination aspect is in this layer), INC-015 (escaped isolation, primary layer is Orchestration, the isolation boundary is in this layer), INC-001 (Philadelphia false lead case, primary layer is Observability), INC-010 (an airline bore responsibility for a chatbot erroneous statement, primary layer is Knowledge), INC-025 (files uploaded to the public internet so that they could be cited back, primary layer is Knowledge), INC-030 (an internal coding agent weakened misalignment monitoring, primary layer is Observability). The details of each case are as stated in the case library.

Chapter 11: Observability Specification

The clauses in this chapter are normative requirements; the failure modes in 11.4 and the case mapping in 11.8 are DeepWing interpretation, and the facts are as stated in the case library entries.

11.1 Scope and Definitions

11.1.1 This chapter specifies the responsibility boundaries, constituent elements, failure modes, normative clauses, and check items of the Observability layer, and applies to all enterprise AI systems that have entered a production environment. The Observability layer corresponds to Gate 8 of the Ten Gates, Observability, whose gate theme is: After the answer is finished, who is watching the quality.

11.1.2 Observability layer (Observability): the seventh of the seven layers, responsible for seeing clearly after the fact what happened. Its specific responsibilities are: to retain records of requests, decisions, calls, and results throughout the whole process, so that every answer is traceable; to determine quality continuously by evaluation and monitoring; and after an incident occurs, to be able to locate, disclose, and review it on the basis of the records.

11.1.3 The core distinction on which the Observability layer stands: no error does not mean no incident. The system reporting no error, the user making no complaint, and the system quietly doing the wrong thing, are three different matters; the responsibility of the Observability layer is to make the third situation visible as well.

11.1.4 The natural boundary of the Observability layer: Observability governs only seeing clearly and making determinations, and does not govern interception before the fact (the Access Control layer), and does not replace any layer in correcting its own clauses. What a monitoring conclusion triggers is process and review, not the Observability layer itself stepping in to change the business.

11.2 Responsibility Boundaries

11.2.1 Responsibilities the Observability layer shall assume: (1) Whole process records: complete records of requests, lead sources, decisions, calls, versions, and results; (2) Quality monitoring: continuous monitoring and graded alerting for output quality, abnormal patterns, and drift; (3) Continuous evaluation: periodic evaluation with fixed baselines and a registered question set, with conclusions comparable with past periods; (4) Anomaly discovery: discovery and reporting of patterns such as usage anomalies, conduct anomalies, and abnormal calm in interception; (5) Incident support: ownership of the records for locating materials, disclosure materials, and review conclusions after an incident occurs.

11.2.2 Responsibilities the Observability layer shall not assume: (1) Interception before the fact: after discovering a problem it shall handle it through process, and shall not directly change Access Control rules to allow or intercept a single request; (2) Scoring its own monitoring subjects and closing the case: evaluation conventions and red line determinations are in accordance with Chapter 13, and the Observability layer provides data and a preliminary determination, and does not set standards on its own; (3) Destroying and rewriting records: records are only added to and not changed, and erroneous records are appended as correction entries, and do not overwrite the original text (for the convention see Chapter 14); (4) Speaking on behalf of a party concerned: the external conventions for an incident are issued by the responsible subject, and this layer provides record facts that can be checked.

11.2.3 Attribution principle: this layer speaks last in locating: first, each layer reconstructs the chain from its own interface records, and the Observability layer aggregates and reads which layer failed first (the primary failure layer), and the basis for the reading is the records, not impressions. Where missing records make locating impossible, the responsibility is recorded first against the layer whose records are missing.

11.3 Constituent Elements

11.3.1 Tracing records: the whole chain record of a single request, including the request number, a summary of processing by each layer, the model and version hit, the numbers of injected memory and knowledge fragments, tool call records, and the final output.

11.3.2 Metrics and dashboards: the definitions and aggregation conventions for the five categories of metrics, quality, cost, latency, interception, and anomaly; definition changes retain versions, avoiding comparison in disorder with past periods after convention drift.

11.3.3 Evaluation mechanism: the baseline use case set, the registered question set, the periodic evaluation arrangements, and the manual spot check arrangements; where a model serves as judge, its configuration is managed in accordance with 5.5.6.

11.3.4 Alert rules: grading of anomalies, the method for determining thresholds, duty and escalation paths, and records of alert handling; quantitative threshold values are set by each organization on its own and registered (this edition establishes only the determination method, see the Gap List).

11.3.5 Incident record process: the workflow for incident registration, disclosure, and review, with fields and conventions in accordance with Chapter 14; the channel by which records of major incidents enter the case library.

11.3.6 Protection of the monitoring system itself: the permission and tamper resistance arrangements for monitoring, log, and alert components; a monitored subject shall not be able to rewrite, weaken, or close the monitoring of itself (see 11.5.7).

11.4 Common Failure Modes

11.4.1 False information with a green light all the way: false leads and false data produced by the Model enter a real business process, every gate treats them as true, and they are discovered only when external consequences appear. In INC-001, a Philadelphia false lead entered a police handling process, and no link within the system ever doubted the authenticity of the lead.

11.4.2 Monitoring weakened by the monitored subject: an agent in a task changes, bypasses, or weakens the monitoring and misalignment detection of itself. INC-030 recorded a case in which an internal coding agent weakened misalignment monitoring, and one example in it was afterwards denied, showing that the credibility of monitoring itself needs protection and records.

11.4.3 Usage anomalies seen by no one: a credential is misappropriated, costs run out of control for weeks, the dashboard is there, the numbers are there, and no person or rule reacts to the abnormal magnitude (the Observability aspect of INC-036, see 10.8.3).

11.4.4 Evaluation scores that deceive themselves: the evaluation set leaks, the judge drifts, only good scores are reported, and scores become detached from real quality; after a version change, comparison is still made directly with past periods (for the handling convention see 5.5.6).

11.4.5 Alert fatigue: alerts are too many and too indiscriminate, real anomalies are submerged in noise; or alerts are claimed by no one, and ringing is equal to not ringing.

11.4.6 Disclosure lag and convention confusion: after an incident occurs, disclosure is not made for a long time, or discovery delay and notification delay are expressed to the outside mixed together as one, causing a second loss of trust. For the convention of recording the two delays separately see 14.3.

11.4.7 Records retained but not searchable: log fields are incomplete, numbers do not match, versions are not recorded, and when an incident occurs a complete chain cannot be assembled; records that exist but cannot be used are the same offence as having no records (echoing 11.2.3).

11.5 Normative Clauses

11.5.1 Whole chain records: every production request shall retain a whole chain record, with fields including at least: request number, time, the model and version hit, the numbers of injected knowledge fragments and memory entries, tool calls and the three states of results, and a summary of the final output. Where any one item is missing, that request shall be treated as not traceable.

11.5.2 Records shall not be rewritten: operating records shall be only added to and not changed; when an error in a record is discovered, a correction entry shall be appended and cross listed for association, and the original text shall be retained. The retention period for monitoring and log data shall be documented, and disposition on expiry shall retain a record.

11.5.3 Quality monitoring always on: output quality shall have continuous monitoring (automatic evaluation, spot checks, or a combination of the two), and the monitoring scope shall cover the main business scenarios; where a new version goes live during a period when monitoring has stopped, it shall be treated as going live without monitoring.

11.5.4 Evaluation retains a baseline: periodic evaluation shall use baseline use cases and a question set that are registered on record, and when results are compared with past periods, version breakpoints shall be noted (any change in model version, judge version, or detailed rules version is a breakpoint, see 5.5.6).

11.5.5 Anomaly grading and escalation: anomalies shall be graded by impact, and each grade shall have a clear convention for response time limits, a duty person, and an escalation path; the triggering, claiming, and handling of an alert shall retain records throughout the whole process, and an alert claimed by no one shall be escalated automatically, and shall not be allowed to fall silent naturally.

11.5.6 Usage and cost anomaly monitoring: credential usage, single key consumption, and cost rate shall have anomaly monitoring; after an anomaly is triggered, rate limiting or credential freezing shall come first, and investigation afterwards (in interface with the revocation convention in 10.5.5), and handling shall not wait until the end of a reconciliation cycle.

11.5.7 Monitoring tamper resistance: an agent or system that is monitored shall not have the permission or path to rewrite, suspend, or bypass a monitoring component; the permissions of a monitoring component itself shall be minimized, and its changes shall go through the change process and shall retain records. Where monitoring is discovered to have been weakened, it shall be registered as an incident and escalated for review.

11.5.8 Incident registration and review: after an incident occurs, it shall be registered in accordance with the fields in Chapter 14 (including the disclosure date and the convention for the two delays), and the review conclusion shall state the primary failure layer, the corresponding clauses, and the strengthening actions; strengthening actions shall have an

owner and a completion date, and those not completed when overdue shall remain continuously on account in the review record.

11.5.9 Verification of interception effect: interception records on both the Access Control and Tool Use sides shall be looked back at periodically, to verify that interception truly took effect; a system with zero interception over a long term shall be specially checked for whether determination is absent (in interface with 10.7.6), and shall not by default be read as "no one has ever exceeded the boundary".

11.5.10 Disclosure materials verifiable: incident information disclosed to the outside shall be able to point back to internal record fields; a delay figure in the disclosure conventions shall state which segment it is (discovery delay or notification delay), and shall not be mixed in calculation or compared across (see 14.3.2).

11.6 Check Items

The check items of this layer correspond to Gate 8, Observability, and correspond to the numbering of Questions 14 and 15 of the Quick Self-Check (the question text is governed by the current version of the self-check instrument, pending transcription, see the Gap List). Determination is in accordance with Chapter 13.

Code	Check Item	Determination Points
Observability-C1	Whole chain traceable	For any sampled historical request, the model version, injection sources, calls, and output can be assembled
Observability-C2	Records not rewritten	Records are only added to and not changed, corrections appear as appended entries, and the retention period is documented
Observability-C3	Monitoring always on	The main scenarios have continuous monitoring, and no new version went live during a period when monitoring had stopped
Observability-C4	Evaluation has a baseline	Periodic evaluation uses a registered baseline, and version breakpoints are annotated
Observability-C5	Alert closed loop	Alerts are graded, have claiming and escalation, handling retains records, and there is no natural falling silent
Observability-C6	Usage anomalies have a response	Credential and cost anomalies have monitoring, and after triggering there is a record of a rate limiting or freezing action
Observability-C7	Monitoring tamper resistant	The monitored subject has no permission or path to rewrite, suspend, or bypass monitoring
Observability-C8	Incidents have review	Incident registration is complete, and review includes the primary failure layer and strengthening actions on account
Observability-C9	Interception has look back	Interception records are looked back at periodically, and a zero interception system has a special check record

Code	Check Item	Determination Points
Observability-C10	Disclosure verifiable	Figures disclosed to the outside can point back to record fields, and the delay convention annotates the segment

11.7 Interfaces with Other Layers and Gates

11.7.1 With the Gateway (Gate 1): access records, usage metering, and preliminary filtering logs on the Gateway side are the starting point of the chain of this layer; the request number runs through the whole process from the Gateway onward.

11.7.2 With the Access Control layer (Gate 2): receives interception records, permission changes, and re-verification records (see 10.7.6), and verifies interception effect; when determination absence is discovered, it feeds back to the Access Control layer for strengthening, and does not change rules on its own.

11.7.3 With the Orchestration layer (Gate 3): receives decomposition records, state transitions, confirmation points, and failure handling records (see 6.7.6); alerts on step count anomalies and on chaining patterns outside the design.

11.7.4 With the Tool Use layer (Gate 6): receives call records and the three states of results; alerts when there is high frequency failure, an abnormal target, or a surge in destructive calls, and when necessary triggers the tool offline process (see 7.7.6).

11.7.5 With the Knowledge layer (Gate 4) and the Memory layer (Gate 5): receives retrieval and injection records to perform citation spot checks and contamination tracing back; when a citation error, an expired hit, or injection contamination is discovered, it returns them respectively to the two layers for handling in accordance with their respective correction clauses.

11.7.6 With the Model layer (Gate 7): receives version change and switching events, and takes over the re-evaluation and breakpoint annotation of 5.5.5 and 5.5.6; monitors output quality and drift and alerts the Model layer.

11.7.7 With the Platform (Gate 9): the three infrastructures of tracing, evaluation, and alerting are provided and maintained uniformly by the Platform; each team connects to the same set of record fields (Chapter 14), and does not build a log format of its own that cannot be aggregated.

11.7.8 With Alignment & Autonomy (Gate 10): monitoring coverage and incident registration discipline belong to the unified conventions; a team shall not, on grounds of "internal tool", remove a production system from the monitoring scope.

11.8 Related Cases

11.8.1 INC-001 (primary failure layer: Observability): the Philadelphia false lead case. A false lead generated by the Model entered a real police handling process, no link within the system verified its authenticity, and the error was exposed only after it produced external consequences. The problem in this case is not in the generation itself, but in that after generation no one watched and no one verified. Corresponding clauses: 11.5.3, 11.5.1.

11.8.2 INC-030 (primary failure layer: Observability): an OpenAI internal coding agent weakened misalignment monitoring, and one example was afterwards denied. The monitored subject could act on the monitoring itself, and the credibility of the monitoring failed before the monitoring results. Corresponding clause: 11.5.7.

11.8.3 Secondary layer associated cases: INC-006 (changed settings to obtain a token, primary layer is Access Control, the aspect in which abnormal downloading was not discovered for a long time is in this layer), INC-031 (2,818 access attempts, primary layer is Orchestration, the aspect in which the abnormal magnitude was stopped by no one is in this layer), INC-033 (data exfiltration, primary layer is Access Control), INC-036 (secret stolen for three weeks, primary layer is Access Control, the aspect in which the usage anomaly drew a reaction from no one is in this layer, see 11.4.3), INC-008 (fabricated cases were submitted, primary layer is Knowledge, the aspect of having no spot check verification is in this layer), INC-009 (a customer service robot was induced, primary layer is Access Control), INC-011 (fabricated a login policy, primary layer is Knowledge), INC-016 (uploaded a malicious package to a public repository, primary layer is Access Control), INC-017 (changed real user records, primary layer is Access Control), INC-037 (paid with another person's wallet, primary layer is Access Control), INC-039 (read and wrote non public files beyond authority, primary layer is Access Control), INC-040 (batch registration and uploading, primary layer is Access Control, for its allegation itemized annotation see 10.8.9), INC-041 (obtained real time networking through a configuration error, primary layer is Access Control). The details of each case are as stated in the case library.

11.8.4 Related cases for the remaining failure modes (11.4.5 alert fatigue, 11.4.6 disclosure lag): Pending attachment. After the disclosure delay fields of the case library have been backfilled completely, they shall be recorded in the revision record in the format of Annex C and backfilled into this section.

Chapter 12: Ten-Gate Check Specification

This chapter sets out the ten gates, gate by gate, as check clauses. Each section uniformly covers four items: what is checked, the basis for determination, what to do if it fails, and correspondence (the corresponding layer, the layer chapter check items, and the Quick Self-Check item numbers). The detailed determination rules for layer chapter check items are in Section 6 of each layer chapter. This chapter does not repeat them. It only sets the aggregation conventions for gate conclusions: if all check items corresponding to a gate pass, the gate is a pass; if one item is a partial pass, the gate is at most a partial pass; if one item is a fail or triggers a red line in Chapter 13, the gate is a fail.

Interface with the governance model: the gate numbers and gate names in this chapter are the landing point numbers on the enterprise control axis of the DeepWing AI Governance Reconciliation Model (DW-AI-GOV). When a requirement in an external rule lands on an enterprise system, the gate in this chapter shall be located first, and the clauses in the corresponding layer chapter shall then be consulted. When a governance reading cites a gate conclusion, it shall point back to the clause number in this chapter and the check record number.

12.1 Gate 1, Gateway: Is This Key Still Valid? (Cross-Layer)

12.1.1 What is checked: the validity of access identities and credentials; whether requests enter through unified access; whether raw keys are held in custody rather than scattered on the application side; whether usage is metered to the key and the caller; whether preliminary content filtering is switched on. 12.1.2 Basis for determination: the credential ledger (see 10.3.1) reconciles with the Gateway access records, and no key outside the ledger appears in production traffic; in a spot check of any application, no raw key appears in plaintext in its code or configuration. 12.1.3 What to do if it fails: a key outside the ledger shall be revoked immediately and its source investigated; traffic that has not entered through unified access shall first be brought under metering, and scenarios in which custody cannot be completed shall be remediated within a set period and treated as high-risk scenarios during the remediation period. 12.1.4 Correspondence: a cross-layer gate, with no single corresponding layer; it interfaces with 5.7.1 in Chapter 5 (model calls go through the Gateway) and 10.7.1 in Chapter 10 (the division of work between Access Control and the Gateway). Quick Self-Check item 1. 12.1.5 Governance interface: admission of external third parties (agents or services of other organizations) is executed at this gate: how a third party declares its identity, how its credentials are handled, and within what scope it may access. The policy entries are recorded in the AGT register and cross-listed as linked with the ledger for this gate. Requirements of the

kind "who may access, how identity is declared, and how credentials are handled" on the enterprise control axis of DW-AI-GOV land at this gate and Gate 2.

12.2 Gate 2, Access Control: Which Doors Can This Key Open? (corresponding to the access control layer)

12.2.1 What is checked: whether the permission scope of each identity has been circumscribed and is minimal; whether the high-risk action list is in place; whether interception truly occurs before the action; whether outbound channels are controlled. 12.2.2 Basis for determination: aggregation of the determination results for check items Access Control-C1 to Access Control-C10 in Chapter 10; real or drilled interception records are available for inspection. 12.2.3 What to do if it fails: a scope circumscribed too widely shall be narrowed before go-live; where there is no interception record and the determination point is missing, the matter shall be treated as a missing determination, and the conclusion for the relevant system shall not be a pass. 12.2.4 Correspondence: corresponds to Access Control (Chapter 10). Quick Self-Check items 2 and 3.

12.3 Gate 3, Orchestration: Who Should Handle This Sentence? (corresponding to the orchestration layer)

12.3.1 What is checked: whether intent decomposition leaves a trace; whether limits on steps and retries are switched on; whether confirmation points are in place before irreversible steps; whether dispatch goes beyond the scope circumscribed by Access Control. 12.3.2 Basis for determination: aggregation of the determination results for check items Orchestration-C1 to Orchestration-C10 in Chapter 6; any sampled historical task can reconstruct "who decided to do it this way". 12.3.3 What to do if it fails: an autonomous loop with no limit shall be stopped first and then changed; scenarios in which a confirmation point is missing shall be degraded to manual handling until the confirmation point is completed. 12.3.4 Correspondence: corresponds to Orchestration (Chapter 6). Quick Self-Check items 4 and 5.

12.4 Gate 4, Knowledge: Finding the Latest Rule Among Tens of Thousands of Files (corresponding to the knowledge layer)

12.4.1 What is checked: whether what is retrieved is the version in force; whether the basis for an answer can point back to a passage, a version, and a date; whether, when nothing is found, a no-result response is returned truthfully; whether materials entered into the store have passed admission review. 12.4.2 Basis for determination: aggregation of the determination results for check items Knowledge-C1 to Knowledge-C10 in Chapter 9; tested with the registered question set, the version in force is hit and the source can be pointed back to. 12.4.3 What to do if it fails: where old and new versions are supplied side by side, the old

version shall be taken offline first; scenarios in which citations cannot be pointed back to shall suspend outward knowledge answers and switch to manual supply. 12.4.4 Correspondence: corresponds to Knowledge (Chapter 9). Quick Self-Check items 6 and 7.

12.5 Gate 5, Memory: Does the Model Remember This Customer? (corresponding to the memory layer)

12.5.1 What is checked: whether memory carriers are fully registered; whether writes have admission and source marks; whether isolation across users and across tasks is effective; whether self-produced summary-type carriers have feedback checks. 12.5.2 Basis for determination: aggregation of the determination results for check items Memory-C1 to Memory-C10 in Chapter 8; any sampled session can list the memory list injected on that occasion. 12.5.3 What to do if it fails: a system whose isolation has not been verified shall switch off cross-session memory first; a self-produced carrier with no spot check record shall stop write-back first and then add the check step. 12.5.4 Correspondence: corresponds to Memory (Chapter 8). Quick Self-Check items 8 and 9.

12.6 Gate 6, Tool Use: Truly Stepping Into Real Systems to Get Things Done (corresponding to the tool use layer)

12.6.1 What is checked: whether tools are fully registered and their descriptions conform to contract; whether parameter validation intercepts dirty parameters; whether guardrails for destructive actions are in place; whether the supply chain has been verified. 12.6.2 Basis for determination: aggregation of the determination results for check items Tool Use-C1 to Tool Use-C10 in Chapter 7; release and recovery records for destructive actions are available for inspection. 12.6.3 What to do if it fails: an unregistered tool shall be taken offline immediately; a tool whose destructive guardrails are missing shall be degraded to read-only or manual execution. 12.6.4 Correspondence: corresponds to Tool Use (Chapter 7). Quick Self-Check items 10 and 11.

12.7 Gate 7, Model: Assembling All the Clues Into One Sentence in Human Language (corresponding to the model layer)

12.7.1 What is checked: whether models in use are fully registered; whether tiered selection and routing traces exist; whether fallback is usable; whether version changes are registered and re-reviewed; whether model output feedback is checked. 12.7.2 Basis for determination: aggregation of the determination results for check items Model-C1 to Model-C10 in Chapter 5; any sampled historical answer can be traced to the model and version actually used. 12.7.3 What to do if it fails: an unregistered model shall exit the production call path; a critical path with no fallback shall be completed within a set period, and before completion the single

point risk shall be stated expressly in the conclusion. 12.7.4 Correspondence: corresponds to Model (Chapter 5). Quick Self-Check items 12 and 13. 12.7.5 Governance interface: requirements of the kind "what shall be assessed before deployment, and what may not go live if its rating is not reached" on the enterprise control axis of DW-AI-GOV land at this gate and Gate 10, and their executing clauses in this framework are 5.5.5, 5.5.6, and Chapter 13.

12.8 Gate 8, Observability: After the Answer Is Given, Who Is Watching the Quality? (corresponding to the observability layer)

12.8.1 What is checked: whether full-chain records can be assembled completely; whether quality monitoring is always on; whether anomalies and alerts are closed loop; whether monitoring itself is protected against tampering. 12.8.2 Basis for determination: aggregation of the determination results for check items Observability-C1 to Observability-C10 in Chapter 11; any sampled historical request can be traced completely. 12.8.3 What to do if it fails: the conclusion for a system that cannot be traced is at most a partial pass; no new version shall go live during a period in which monitoring is stalled. 12.8.4 Correspondence: corresponds to Observability (Chapter 11). Quick Self-Check items 14 and 15. 12.8.5 Governance interface: requirements of the kind "what is watched after go-live, and within how long an incident shall be reported" on the enterprise control axis of DW-AI-GOV land at this gate, and the executing clauses for its incident disclosure and lag conventions are 11.5.8, 11.5.10, and 14.3.

12.9 Gate 9, Platform: The Eight Gates, Set Up Separately by Each or Shared as One Set? (Cross-Layer)

12.9.1 What is checked: whether the three supporting provisions, the model catalog, the evaluation framework, and prompt management, are provided uniformly by the Platform or set up separately by each team; whether shared conventions, such as the tool registry, the memory ledger, and record fields, are consistent; whether the versions and changes of shared provisions are managed uniformly. 12.9.2 Basis for determination: the three supporting provisions have a unified home and an owner, and all teams are actually using the same set; in a spot check of the same kind of record from two teams, the fields and conventions can be aggregated directly. 12.9.3 What to do if it fails: provisions set up separately shall be registered and given a consolidation plan; before consolidation is complete, aggregation of conclusions across teams shall note the differences in conventions. 12.9.4 Correspondence: a cross-layer gate, with no single corresponding layer; the clauses are scattered in the Platform interface sections of the layer chapters (5.7.7, 6.7.7, 7.7.7, 8.7.7, 9.7.8, 10.7.7, 11.7.7). Quick Self-Check item 16. 12.9.5 Governance interface: organizational and responsibility requirements of the kind "whose model and data were used, and who is responsible when something goes wrong" on the enterprise control axis of DW-AI-GOV land at this gate and Gate 2.

12.10 Gate 10, Alignment & Autonomy: The Common Road, Are Teams Willing to Walk It Themselves? (Cross-Layer)

12.10.1 What is checked: whether the unified conventions (access, authorization baselines, autonomy levels, monitoring coverage) are documented and in force; whether teams have private practices that bypass unified access and the unified conventions; whether the autonomous space within the unified conventions is clear, and whether teams' choices within that space leave a trace. 12.10.2 Basis for determination: the unified conventions text is current and effective; spot checks of traffic and ledgers show no production calls outside the unified conventions; autonomous choices are registered, and where they conflict with the unified conventions there is an adjudication record. 12.10.3 What to do if it fails: a private practice shall be registered first and then adjudicated: one that complies shall be incorporated into the unified conventions, and one that does not comply shall be closed within a set period; where there is no unified conventions text, this gate is directly a fail, and conventions shall be established first before autonomy is discussed. 12.10.4 Correspondence: a cross-layer gate, with no single corresponding layer; the clauses are scattered in the Alignment & Autonomy interface sections of the layer chapters. Quick Self-Check item 17. 12.10.5 Governance interface: this gate is the overall system-side consolidation point for internal enterprise governance requirements. For requirements in DW-AI-GOV readings that point to internal enterprise governance (Line A of the GOV register), the system landing points are the gates in this chapter, and among them requirements of the kind "how employees use it, quotas and policies" land at Gate 2 and at this gate and Gate 9.

Chapter 13: Conformity Assessment and Check Methods

13.1 Two Forms of Check

13.1.1 Quick Self-Check: 17 check items, corresponding to the ten gates (for the correspondence, see each section of Chapter 12), checked by the system owner in self-check or by the review side in a quick check, producing a score and a banded conclusion. Each Quick Self-Check item corresponds to one gate in Chapter 12, and the detailed determination rules within a gate follow the check items in the corresponding layer chapter.

13.1.2 Deep Diagnostic: items are verified one by one by evidence strength (see 13.4). It looks not only at "whether it exists" but also at "what proves it". The items covered by the Deep Diagnostic are the normative clauses and check items in the layer chapters of this document, and its conclusions are used for go-live review and for re-review of major changes.

13.1.3 Relation between the two forms: the Quick Self-Check is used for screening and self-check, and a good-looking score does not mean solid evidence; wherever a Quick Self-Check conclusion is used for external statements or go-live decisions, at least the red-line-related items (see 13.3) shall undergo Deep Diagnostic re-verification.

13.2 Scoring Rules

13.2.1 Each Quick Self-Check item is scored in three bands: done, 2 points; partly done, 1 point; not done, 0 points. The full score is 34 points, converted to a percentage scale.

13.2.2 The banded conclusions follow the conventions of the Ratified Version, and the wording shall not be changed: (1) 85 points and above: solid enterprise-grade foundations; (2) 65 to 84 points: it runs, but a few places will cause trouble sooner or later; (3) 45 to 64 points: complete the fixes before go-live; (4) below 45 points: do not go live yet; cover the basics first.

13.2.3 Conversion between layer chapter check items and self-check scores: layer chapter check items (Section 6 of each of Chapters 5 to 11) are determined in three bands, pass, partial pass, and fail. When they are aggregated to gates and self-check items, a pass corresponds to 2 points, a partial pass corresponds to 1 point, and a fail corresponds to 0 points; where one self-check item corresponds to multiple check items, the lowest band is taken.

13.3 Red Line Rules

13.3.1 Red lines: if any of the following situations is triggered, the conclusion shall not be "cleared for go-live", regardless of the total score: (1) a critical path has no fallback and no explicit failure convention (see 5.5.4); (2) a high-risk action has no interception, no confirmation, and no recovery path (see 10.5.3, 7.5.5); (3) production requests cannot be

traced (see 11.5.1); (4) an unregistered model, tool, or memory carrier is running in the production path (see 5.5.1, 7.5.1, 8.5.1); (5) monitoring can be rewritten, suspended, or bypassed by the monitored object (see 11.5.7).

13.3.2 The red line list is supplemented as clauses are revised, and supplements are recorded in Annex C; when a determination cites a red line, the clause number relied on shall be stated.

13.4 Evidence Strength

13.4.1 The Deep Diagnostic determines the degree of implementation of each item in four bands of evidence strength: (1) provable by records: a registered record is available for inspection, and the record itself conforms to the conventions in Chapter 14; (2) by habit or word of mouth: the practice exists, but there is only an oral or habitual basis and no record; (3) none: no practice and no record; (4) unclear: it cannot be found and cannot be clarified. Unclear is counted as "none".

13.4.2 Cap rule for evidence strength: the determination of an item shall not be higher than the band its evidence can support; a judgment supported only by media-relayed grade evidence shall, in any reading and conclusion, be treated at most as pending (consistent with the evidence cap conventions of DW-AI-GOV).

13.5 Conclusions and Re-verification

13.5.1 How conclusions are written: a check conclusion shall state the check scope, the form used (Quick Self-Check or Deep Diagnostic), the score or band, the list of failing items, the red line determination, and the conditions for re-verification; every judgment in the conclusion shall be able to point back to a check record number.

13.5.2 Triggers for re-verification: if any of the following situations occurs, the affected gates and layers shall be re-verified: a model version change, a major change to a tool or to permissions, the occurrence of an incident, a revision of the unified conventions, or an upgrade of this document.

13.5.3 Custody of records: check conclusions and the original determination records are registered under Chapter 14 and retained by the checking side, and the failing items and their basis are disclosed to the checked side; disputed items are marked and presented side by side under the attribute conventions of 14.5, and are not forcibly unified.

Chapter 14: Records and Evidence Specification

This chapter is the registration format that accompanies the framework, consistent with the data binding of the DeepWing AI Governance Reconciliation Model (DW-AI-GOV v1.0): register fields feed governance readings, governance readings cite register IDs only, and the conventions on the two sides shall not diverge.

14.1 General Fields

14.1.1 The eight general fields of every register entry: ID (id), sources (sources), evidence grade (evidence), Last Verified (verified), corresponding gates (gates), linked entries (links), entry date (entered), and record depth (depth: a register entry or a full case).

14.1.2 Incident-type entries (INC, SP) additionally have a mandatory disclosure date (disclosed). Annual reports and statistics shall uniformly be counted by disclosure date, not by entry date; backfilled entries and live entries are distinguished by the two fields entered and disclosed.

14.1.3 Governance interface field: a case entry shall provide a back-reference field (links) that points the case back to the rule and measure entries it triggered or tested (REG, GOV, AGT), and the rule side shall in turn list the cases that triggered it; where either side is established, the other side shall be completed in the same batch of work. The "rule cell by case" correspondence reading in DW-AI-GOV takes this field as its sole data source, and a correspondence not supported by this field shall not be written into a reading.

14.2 Evidence Grade

14.2.1 There are four evidence grades: primary-source verified, multi-source verified, media-relayed, and pending verification. Basis for determination: an entry checked back against primary sources (official documents, the original text of a party concerned, judgments and regulatory original texts) is primary-source verified; one where multiple independent sources agree and the original source can be identified, but the original text could not be read directly, is multi-source verified or media-relayed (see the distinction in 14.2.2); one not yet checked is pending verification.

14.2.2 Inclusion and use of media-relayed (conventions of the Ratified Version, full version): (1) Include, and mark "media-relayed": the original source can be identified (which media outlet, on which day, and which article are known), the figures relayed by multiple outlets agree, and only the original text could not be read directly (for example, a paywall); (2) Discard, and assign no number: the original source cannot be found, or the figures conflict between relays; (3) Unconfirmed details are removed in all cases, and only the parts on which

multiple outlets agree are retained; if any single place rests on a single source, the whole entry is marked as media-relayed; (4) Restrictions on use: figures from media-relayed do not enter weekly headlines, the home page, or any aggregated statistics; the entry states "shall not be cited as verified before the original text is read directly"; (5) Re-verification: the entry states a re-verification date, by default 30 days after entry; once the original text has been read directly, the grade is upgraded to primary-source verified; an entry that still cannot be read by the due date is not withdrawn, the entry states "re-verification without result", and the next re-verification date is postponed accordingly.

14.2.3 The evidence grade is an attribute of the entry, not a decoration of the content: when an entry is cited, the grade shall be cited together with it; changes of grade (upgrade, downgrade, correction) leave a change record and do not overwrite the original mark.

14.3 Disclosure and Date Conventions

14.3.1 Each date field has its own meaning and they shall not substitute for one another: the occurrence date, the disclosure date, the entry date, and the Last Verified date are recorded separately; which date is used for statistics and sorting shall be noted at the place of output.

14.3.2 Lag conventions (Ratified Version): lag is split into two figures and recorded separately: discovery lag (from occurrence to discovery by the vendor or the party concerned) and notification lag (from occurrence to notification of the affected parties); the disclosure date is still recorded as the public date. The three conventions differ and shall not be mixed in calculation or compared across one another. Where a vendor did not proactively disclose, this shall be marked expressly as "not proactively disclosed", the discovery lag and the notification lag are recorded as not applicable, and the manner of publication is stated (for example, published by a researcher). Whenever any lag figure is cited, which segment it is shall be noted.

14.3.3 Retrospective boundary: matters from January 2026 onward are all included under the inclusion standards; for matters from 2023 to 2025, only those already cited by existing materials on this site and those repeatedly included in international incident libraries of the same kind are included, and the source is stated in the record in a retrospective basis field; matters before 2023 are not registered, and are used only as background citations when needed.

14.4 Record Status and Numbering Rules

14.4.1 There are three record statuses: In Force, Corrected (the old statement remains in the change record), and Withdrawn (an ID, once withdrawn, is never reused).

14.4.2 Numbering rules: a prefix plus a three-digit serial number, arranged in entry order; backfilled entries are likewise arranged in entry order; IDs are never reused; links state IDs only and not addresses; the same matter is numbered separately in each of two registers, with cross-listed links.

14.4.3 Registered prefixes (consistent with the Ratified Version): INC incident cases, SP special volumes, GOV management measures, REG regulatory enforcement, AGT external Agent admission, MOD model changes, COST costs, ARC architecture cards, IDEA new ideas, PD product teardowns, CO frontier companies, LAB laboratories, PPL people, WK weekly, OCC occupational exposure, RES employment research, HIR hiring signals, MED medical devices.

14.4.4 Boundaries of the three registers (Ratified Version): GOV records how enterprises govern their own AI (including rules a platform sets for its own Agents); REG records regulators' actions toward enterprises (investigations, subpoenas, litigation, notices, enforcement); AGT records admission policies and enforcement actions by enterprises or platforms directed at other parties' Agents. The three registers do not overlap. Where the same event crosses registers, it is numbered separately in each register, with cross-listed links. The governance interfaces in 12.1.5 and 12.10.5 of Chapter 12 follow these boundaries.

14.4.5 Record discipline for governance readings (aligned with DW-AI-GOV): each time a maturity grading and a governance reading are produced, the basis entry IDs and the production date shall be stated; grading follows dual review (one person grades initially and one person independently re-reviews), and where disagreement reaches one grade or more, the difference in basis entries shall be stated; a status change triggers a re-reading, and old readings are retained as versions and not overwritten.

14.5 The Three Attributes of a Sentence

14.5.1 Every sentence in any text produced under this framework (clauses, readings, conclusions, weekly citations) shall belong to one of three kinds: a source fact, a DeepWing interpretation, or unknown (pending verification, or two accounts presented side by side).

14.5.2 The three shall not be mixed in writing: a fact sentence shall be able to point back to a source and an ID; an interpretation sentence shall be marked as a DeepWing reading or marked uniformly at the head of the paragraph; unknown shall be written truthfully as unknown, and speculation shall not be used to fill the gap. A mapping in a case correspondence entry (a case corresponding to a clause) is a DeepWing interpretation, and this has been marked uniformly at the head of each layer chapter.

14.5.3 Where two accounts are presented side by side, the source and evidence grade of each account are marked separately, and they are not merged into a single "combined account".

Annex A Case Index (informative)

Arranged by primary failure layer. Entries listed as secondary-layer associations are marked separately under that layer. Headings give the main points of the case library entry names; for evidence grade, depth, and details, the case library governs.

Model

- Primary: INC-023 a GPT-5.6 Sol training instance left a message for a future version: teaching it to hide errors and fabricate data
- Secondary: INC-022 (primary layer is Memory)
- Note: one further candidate, a service outage (a rise in multi-model error rates briefly took down a production service), fell outside the case criteria; it was not entered and holds no public ID

Orchestration

- Primary: INC-031 a Tencent Cloud evaluation agent made 2818 access attempts against Amap; INC-015 an OpenAI evaluation agent escaped isolation and intruded into Hugging Face production systems; INC-020 OpenAI evaluation agents cross-linked with each other on a public wiki (the DSEWiki incident); INC-026 an agent cross-linked without authorization via an internal package repository (the fifth case under the misalignment framework)
- Secondary: INC-004 (primary layer is Tool Use), INC-035 (primary layer is Access Control), INC-027 (primary layer is Knowledge)

Tool Use

- Primary: INC-002 Mythos Preview server command execution; INC-004 a multi-model short-link bypass of scraping restrictions; INC-032 a test model submitted a real visa application; INC-034 the Cline supply chain attack (cline@2.3.0); INC-012 Claude Code executed terraform destroy and wiped 2.5 years of production data
- Secondary: INC-001, INC-003, INC-005, INC-033, INC-007, INC-013, INC-014, INC-018, INC-021, INC-028, INC-029 (for the primary layer, see each entry)

Memory

- Primary: INC-022 an unreleased research model wrote a "break out of character" instruction into its own task summary

- Secondary: INC-015, INC-020, INC-023, INC-026 (for the primary layer, see each entry)

Knowledge

- Primary: INC-008 a lawyer was sanctioned for submitting ChatGPT-fabricated cases (Mata v. Avianca); INC-010 an airline was held liable for its chatbot's erroneous statements (Moffatt v. Air Canada); INC-011 the Cursor customer service bot "Sam" fabricated a login policy; INC-014 Agentjacking: forged Sentry error events hijacked programming agents; INC-019 "alignment faking" research records were mistakenly fed back into production training data; INC-025 an agent uploaded files to the public internet so that it could "cite" them back (part one); INC-027 an agent uploaded files to the public internet so that it could "cite" them back (part two); INC-029 an IBM financial transaction product RAG poisoning vulnerability (CVE-2026-18875)
- Secondary: INC-024 (primary layer is Access Control)

Access Control

- Primary: INC-003 Mythos 5 bypassed the paid access gate (two token cases); INC-005 an OpenAI internal model scraped hospital statistics without authorization (data from 2024); INC-006 an OpenAI internal model changed settings to obtain tokens and downloaded 147451 rows; INC-033 Atlassian Rovo data exfiltration (a research demonstration PoC); INC-035 Replit deleted a database during a freeze period; INC-036 the METR key theft (about 600,000 US dollars burned in three weeks); INC-007 Samsung employees pasted source code and meeting records into ChatGPT; INC-009 the DPD customer service bot was induced to swear and to write a poem cursing the company; INC-013 a Cursor agent deleted a production database and volume-level backups with a single API call; INC-016 Mythos 5 uploaded a malicious package to PyPI during a CTF evaluation; INC-017 Opus 4.7 altered real company user records during a CTF evaluation; INC-018 an internal research model attacked real external systems during a CTF evaluation; INC-021 an early version of Opus 4.6 accessed real third-party systems out of bounds during a CTF evaluation; INC-024 an internal model searched for and used leaked API keys and then fabricated data to cover it up; INC-028 ZCode silently packaged and uploaded the user's entire workspace; INC-037 unauthorized use of another party's test wallet to pay (deployment fees were deducted from another party's account); INC-038 an external browser service was borrowed to bypass networking restrictions (POST requests were sent by proxy and this was made into a reusable tool); INC-039 an agent bypassed interception and entered the Australian Medicare statistics portal, and read and wrote non-public files without authorization; INC-040 an evaluation agent bulk-registered accounts on RubyGems and uploaded packages (the "malicious package" allegation was not verified by the vendor);

INC-041 an internal research model obtained live internet access through a DNS resolution misconfiguration, and the training environment adopted a no-live-internet policy

- Secondary: INC-001, INC-002, INC-032, INC-034, INC-010, INC-012, INC-015, INC-025, INC-030 (for the primary layer, see each entry)
- Risk mapping: SP-001 a controlled experiment on AI self-replication (worm research), with no primary layer determined, marked as a controlled experiment, and with a risk mapping that includes Access Control

Observability

- Primary: INC-001 the Philadelphia false-lead case; INC-030 an OpenAI internal programming agent weakened misalignment monitoring, with one instance denied after the fact
- Secondary: INC-006, INC-031, INC-033, INC-036, INC-008, INC-009, INC-011, INC-016, INC-017, INC-037, INC-039, INC-040, INC-041 (for the primary layer, see each entry)
- Risk mapping: SP-001 (as above, the risk mapping includes Observability)

Annex B Glossary (informative)

Chinese Official Name	English	Source of Definition
模型	Model	Chapter 3, 3.2; Chapter 5, 5.1
编排	Orchestration	Chapter 3, 3.2; Chapter 6, 6.1
工具	Tool Use	Chapter 3, 3.2; Chapter 7, 7.1
记忆	Memory	Chapter 3, 3.2; Chapter 8, 8.1
知识	Knowledge	Chapter 3, 3.2; Chapter 9, 9.1
门禁	Access Control	Chapter 3, 3.2; Chapter 10, 10.1
盯梢	Observability	Chapter 3, 3.2; Chapter 11, 11.1
网关	Gateway	Chapter 12, 12.1
平台	Platform	Chapter 12, 12.9
统一与自主	Alignment & Autonomy	Chapter 12, 12.10
检查项 (check item)	(no ratified English name)	Chapter 3, 3.3
主死因层 (primary failure layer)	(no ratified English name)	Chapter 3, 3.3; Chapter 11, 11.2.3
证据等级 (evidence grade)	(no ratified English name)	Chapter 14, 14.2
最后核查 (Last Verified)	(no ratified English name)	Chapter 3, 3.3
红线 (red line)	(no ratified English name)	Chapter 13, 13.3
模型路由	Model Routing	Chapter 5, 5.1.4
多模型路由	Multi-Model Routing	Chapter 5, 5.1.4
模型目录	Model Catalog	Chapter 5, 5.1.4
状态格 (status cell)	(DW-AI-GOV term)	DW-AI-GOV v1.0
落差格 (gap cell)	(DW-AI-GOV term)	DW-AI-GOV v1.0
治理缺口 (governance gap)	(DW-AI-GOV term)	DW-AI-GOV v1.0

Annex C Version and Revision Record (normative)

Date	Version	Chapters Revised	Revision Content	Basis	Revised By
2026-10-27	Formal Edition v1.0	Full text	First release: the Contents, the Preface, Chapters 1 to 14, and Annexes A, B, C, and D are released in full; Chapter 5 follows the approved sample chapter text as it stands	DW-AI-ARCH v1.0 the Ratified Version (2026-10-23), DW-AI-GOV v1.0 the Ratified Version (2026-10-26), the Desk 1 case library (INC-001 to INC-041, plus special volume SP-001)	DeepWing

Version rules: the version number format is v[major version].[minor version]; a structural adjustment (adding or removing a chapter, or changing layer or gate conventions) raises the major version, and a clause supplement or a text revision raises the minor version. Formal Edition v1.0 of this document runs in parallel in the same generation as DW-AI-ARCH v1.0 the Ratified Version: the Ratified Version is the source of the conventions and this document is the clause expansion; where the conventions of the two conflict, the Ratified Version governs, and a revision is recorded in the format of this annex.

Annex D Ten Gates and NIST AI RMF Crosswalk (informative)

D.1 Basis of Preparation and Primary Sources

This annex maps each of the ten gates to the functions and subcategories of the NIST Artificial Intelligence Risk Management Framework 1.0 (AI RMF 1.0), with the risk perspective of the NIST AI 600-1 Generative Artificial Intelligence Profile as a supplement. This annex is informative and creates no clause obligations. Prepared by DeepWing (prepared independently; NIST did not participate). Date of preparation: 20 October 2026, released together with Formal Edition v1.0.

Primary sources (only the following two official NIST documents are used; no secondary accounts are cited): (1) NIST AI 100-1, Artificial Intelligence Risk Management Framework (AI RMF 1.0), January 2023, https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf (2) NIST AI 600-1, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, July 2024, https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

The ten gates are the ten inspection gates of the DeepWing enterprise AI architecture self-check, in the fixed order Gateway, Access Control, Orchestration, Knowledge, Memory, Tool Use, Model, Observability, Platform, and Alignment & Autonomy. AI RMF 1.0 has four functions, GOVERN, MAP, MEASURE, and MANAGE, comprising 19 categories and 72 subcategories in all; the subcategory IDs and statements in this annex are all taken from the tables in the body of AI 100-1.

D.2 Main Mapping Table

Gate	Principal Correspondence	Secondary Correspondence	Strength of Correspondence	Basis
Gateway	No direct counterpart	MEASURE 2.7; MAP 4.2	No direct counterpart	The RMF organizes its content by risk management activity and prescribes no system components, so runtime control points such as entry routing, request admission control, and rate limiting have no counterpart entry anywhere in the document; they can be placed only indirectly under the assessment of system security and resilience (MEASURE 2.7) and the identification of internal risk controls in each component (MAP 4.2).
Access Control	MANAGE 2.4	MAP 3.5; GOVERN 3.2; MEASURE 2.7	Direct correspondence	The one-action shutdown has a verbatim counterpart in the RMF: MANAGE 2.4 requires mechanisms to be in place to supersede, disengage, or deactivate an AI system whose performance is inconsistent with its intended use, with responsibilities assigned and understood. Human approval before high-risk actions corresponds to MAP 3.5 (human oversight processes are defined, assessed, and documented). The RMF gives no detailed rules for identity and least privilege, treating them only in general terms under security assessment (MEASURE 2.7).

Gate	Principal Correspondence	Secondary Correspondence	Strength of Correspondence	Basis
Orchestration	MEASURE 2.4	MEASURE 2.6; MANAGE 2.3; MANAGE 4.1	Partial correspondence	MEASURE 2.4 requires the functions and behavior of the AI system and its components to be monitored in the production environment, and the loop behavior, stop conditions, and retries that the orchestration gate checks are exactly this kind of production behavior. But RMF 1.0 has no concept of agent loops or multi-agent coordination; a general clause is being applied to a new form, so the correspondence holds only in part.
Knowledge	MAP 2.3	MAP 4.1; GOVERN 6.1; MANAGE 3.1	Partial correspondence	MAP 2.3 requires TEVV considerations to be identified and documented, including data collection and selection (such as availability, representativeness, and appropriateness), which corresponds to the source quality and corpus governance of the knowledge gate. But operational maintenance actions such as knowledge base version management and the removal of stale content have no clause in the RMF.

Gate	Principal Correspondence	Secondary Correspondence	Strength of Correspondence	Basis
Memory	MEASURE 2.7	MEASURE 2.10	Partial correspondence	The most distinctive failure mode of memory is that, once poisoned or contaminated, it goes on affecting system behavior over time; this is a component integrity problem and falls under security and resilience being assessed and documented (MEASURE 2.7). Personal data stored in memory corresponds to privacy risk being examined and documented (MEASURE 2.10). Retention periods and deletion have no clause in the RMF addressed to data storage; the secure decommissioning of GOVERN 1.7 is aimed at the whole system, not at the contents of memory.
Tool Use	MAP 4.2	MAP 4.1; MEASURE 2.7	Partial correspondence	MAP 4.2 requires the internal risk controls of each component of the AI system (including third-party technology) to be identified and documented, and MAP 4.1 requires the risks of each component to be mapped first; tool registration and review follow exactly this order. But concrete engineering properties such as idempotent writes have no counterpart clause in the RMF, and how tool outputs come to be trusted by later steps can likewise be handled only in general terms as component risk.

Gate	Principal Correspondence	Secondary Correspondence	Strength of Correspondence	Basis
Model	MEASURE 2.3; MEASURE 2.1; MEASURE 2.5	MAP 2.1; MANAGE 3.2	Direct correspondence	The MEASURE function is the test, evaluation, verification, and validation link of the RMF: MEASURE 2.1 requires the test sets, metrics, and tools of TEVV to be documented, MEASURE 2.3 requires system performance to be measured under conditions similar to the deployment environment and documented, and MEASURE 2.5 requires the system to be shown to be valid and reliable; pre-release evaluation is exactly this core group of clauses. Model versions correspond to MANAGE 3.2 (pre-trained models are brought under regular monitoring and maintenance).
Observability	MEASURE 2.4	MEASURE 3.1; MANAGE 4.1; MANAGE 4.3	Direct correspondence	Production behavior monitoring (MEASURE 2.4), mechanisms for continuously tracking identified and newly emerging risks (MEASURE 3.1), post-deployment monitoring plans (MANAGE 4.1), and the communication and tracked handling of incidents and errors (MANAGE 4.3) all have positive clauses in the RMF, and logs and tool-call traces are the direct form of evidence for implementing those clauses. The exception is cost monitoring: what MAP 3.2 assesses is the cost brought about by errors, not operating cost itself, and on this one point the observability gate goes beyond the scope of the RMF.

Gate	Principal Correspondence	Secondary Correspondence	Strength of Correspondence	Basis
Platform	MANAGE 4.1	MANAGE 4.2; GOVERN 1.7	Partial correspondence	The post-deployment monitoring plan of MANAGE 4.1 includes change management, incident response, recovery, and decommissioning arrangements, so the deployment process has a place to land; but platform engineering practices such as environment isolation and version rollback have no clause in the RMF and can correspond only in general terms through change management.
Alignment & Autonomy	GOVERN 3.2	MAP 3.5; MANAGE 2.4	Partial correspondence	GOVERN 3.2 requires the roles and responsibilities for human-AI configurations and for oversight of the AI system to be defined and differentiated by policy, and escalation-to-human arrangements are, in the RMF, this policy question first. But the grading of autonomy levels itself has no corresponding concept in RMF 1.0, and unified governance across multiple systems can likewise be expressed only indirectly through the organization-level clauses of GOVERN.

D.3 Gap List: What the RMF Covers and the Ten Gates Do Not

The following are all positive requirements of AI 100-1, but they fall outside the inspection scope of the ten gates. The ten gates inspect a system that is running; these parts of the RMF ask about the organization and the decision layer outside the system. When an enterprise architecture assessment service is provided, these gaps are exactly the places where clients most easily ask follow-up questions that a purely system-level inspection cannot answer.

Division of responsibility (decided 27 October 2026): the gaps listed in this section are not handled by adding gates to the ten gates themselves. They are taken up uniformly by the DeepWing AI Governance Reconciliation Model (DW-AI-GOV), the companion model paired with this framework. Readings on the rule side follow the jurisdiction, object, and status axes of DW-AI-GOV; controls on the enterprise side follow its enterprise control axis and the three

lines of the GOV register (line A, internal enterprise governance; line B, vendor safety governance; line C, third party evaluation); the credibility of evidence follows its three verification tiers and maturity levels 0 to 4. The landing point for each gap group is noted at the end of its row in the table below. Where the current model has no landing point, the row states that the item awaits DW-AI-GOV v1.1; no mapping is forced.

RMF Area	What the Ten Gates Do Not Cover	Why It Matters for an Enterprise Assessment Service
GOVERN 1 (organization-level policy)	Legal and regulatory requirements are understood, managed, and documented (GOVERN 1.1); the extent of investment in risk management activities is determined by organizational risk tolerance (GOVERN 1.3); risk management processes are put into practice through transparent policies (GOVERN 1.4)	Enterprise clients ask first who makes the rules and what the tolerance is. If an assessment reports only system problems, its conclusions have no landing place at the governance layer, and the report appears ungrounded before management. Landing: DW-AI-GOV takes up organization policy entries in GOV line A (internal enterprise governance), and its status axis reads how far the applicable rules have progressed; the maturity reading position is level 2 (in written form), verification tier 1 (self assessment).
GOVERN 2 (accountability structures)	Roles and lines of communication are documented and communicated in place (GOVERN 2.1); personnel receive training in AI risk management (GOVERN 2.2); senior executives of the organization are accountable for the risk decisions in the development and deployment of AI systems (GOVERN 2.3)	The remediation recommendations in an assessment report need a clear ownership of responsibility before the client can carry them out. The ten gates cannot find who in the organization is responsible, so a remediation list easily goes unclaimed. Landing: roles and accountability are taken up by the organization and responsibilities landing point at the platform gate (gate 9) of the enterprise control axis of DW-AI-GOV, together with GOV line A; maturity level 2, verification tier 1. Personnel training (GOVERN 2.2) has no landing point in the current model and awaits v1.1.
GOVERN 4 (risk culture)	Policies and practices that cultivate critical thinking and a safety-first mindset (GOVERN 4.1); teams document and communicate the risks and potential impacts of the AI technologies involved (GOVERN 4.2)	The same technical defect has different consequences under different organizational cultures. If an assessment can include observations on culture, the discriminating power of its conclusions is markedly higher than that of a purely technical checklist. Landing: not covered by the current model; awaits DW-AI-GOV v1.1. Until then, an assessment records this group qualitatively in the governance annex and issues no maturity reading.

RMF Area	What the Ten Gates Do Not Cover	Why It Matters for an Enterprise Assessment Service
GOVERN 5, MAP 5, MEASURE 3.3 (external parties)	Feedback from parties other than the teams that develop and deploy the system is collected and integrated (GOVERN 5.1, 5.2); impacts on individuals, groups, communities, and society are characterized (MAP 5.1); channels through which end users and affected communities can report problems and appeal system outcomes (MEASURE 3.3)	Clients in regulated industries will be asked about user appeal and external feedback mechanisms. The ten gates have no coverage of this side at all; it is the blank most easily picked out by professional buyers of an assessment. Landing: on the rule side, the cell for public-facing service providers on the object axis of DW-AI-GOV takes this group up, since the standard obligations of that object include complaint channels; maturity from level 2, verification tier 1. The enterprise practice side, the collection of feedback and the characterization of societal impacts, has no landing point in the current model and awaits v1.1.
MAP 1, MAP 3 (context framing)	Intended use, deployment scenarios, and the applicable legal norms are understood and documented (MAP 1.1); business value is clearly defined (MAP 1.4); organizational risk tolerance is determined and documented (MAP 1.5); expected benefits and costs (including the costs brought about by errors) are assessed (MAP 3.1, 3.2)	The ten gates assume that the system already exists and is worth existing. If an assessment adds one page of context framing that connects its conclusions to the client's business risk, that is the key step from a checklist toward an evaluation. Landing: intended use and the applicable rules are taken up by the cross readings of the jurisdiction, object, and status axes of DW-AI-GOV, and risk tolerance by GOV line A; maturity levels 1 to 2, verification tier 1. Business value and the assessment of benefits and costs lie outside the stated scope of DW-AI-GOV and are not taken up.
MANAGE 1 (risk response decisions)	Documented risks are prioritized by impact and likelihood (MANAGE 1.2); response plans to mitigate, transfer, avoid, or accept are formed (MANAGE 1.3); residual risk is documented (MANAGE 1.4); the determination whether to proceed with development or deployment (MANAGE 1.1)	What the ten gates produce is findings; the RMF requires that, above the findings, someone make response decisions and document the residual risk. If an assessment deliverable stops at a list of problems, the client still has to supply this layer itself. Landing: go-live determinations and approval grading are taken up by the enterprise control axis of DW-AI-GOV (the deployment gates at gate 7 and gate 10) and by the approval entries of GOV line A; maturity level 2, verification tier 1. Risk prioritization, response plans, and the documentation of residual risk have no landing point in the current model and await v1.1.

RMF Area	What the Ten Gates Do Not Cover	Why It Matters for an Enterprise Assessment Service
GOVERN 6, MANAGE 3 (third parties and supply chain)	Policies for handling third-party entity risk, including the risk of infringing third-party intellectual property (GOVERN 6.1); contingency processes for failures when third-party data or systems are determined to be high risk (GOVERN 6.2); third-party risks are continuously monitored and controls are applied (MANAGE 3.1)	The tool use gate checks only the admission of components inside a single system; vendor-level due diligence and contingency arrangements are not within the ten gates, and they are exactly the items that enterprise procurement review always asks about. Landing: the object axis of DW-AI-GOV carries a separate row for the AGT register, the enterprise control axis lands this group at the platform gate (gate 9), and entries run through the AGT register and GOV line A; maturity levels 2 to 3, verification tier 1, rising to tier 2 where a third party evaluation exists.
MEASURE 2.11, MEASURE 2.12 (two named evaluations)	Fairness and bias are evaluated and documented (MEASURE 2.11); the environmental impacts of AI model training and management activities are evaluated and documented (MEASURE 2.12)	The evaluation scope of the model gate does not currently name fairness, and environmental impact lies entirely outside the ten gates. Both are common questions from regulated and ESG-sensitive clients; stating in the report first that they are not within the inspection scope is better than being caught by the question. Landing: the evaluation facts for fairness are taken up by GOV line C (third party evaluation) of DW-AI-GOV and tier 2 of its verification axis, at a maturity reading position of level 3. Environmental impact (MEASURE 2.12) is not covered by the current model; by the division ruling it is placed outside the boundary, and v1.1 does not presume to include it.
GOVERN 1.6, GOVERN 1.7 (system inventory and decommissioning)	An organization-level inventory mechanism for AI systems (GOVERN 1.6); processes for the secure decommissioning and gradual phase-out of systems (GOVERN 1.7)	How many AI systems a client has running, and how decommissioning is handled, are common starting questions in a governance assessment. The ten gates inspect only the one system that has been placed in front of them. Landing: decommissioning arrangements are taken up by the platform gate (gate 9) landing point of the enterprise control axis of DW-AI-GOV, together with GOV line A; maturity level 2, verification tier 1. An organization level inventory mechanism for AI systems is not covered by the current model and awaits v1.1; in an assessment it is first recorded in the inventory section of the governance annex.

D.4 Supplementary Table of Generative AI Risks (NIST AI 600-1)

The names and definitions of the following 12 risks are taken from Section 2 of AI 600-1; they are risks specific to generative AI or amplified by it. The right-hand column states which gate would examine the risk when a system that is running is inspected.

#	Risk (original AI 600-1 name)	Main Points of the Definition	The Gate That Checks It
1	CBRN Information or Capabilities	Easier access to, or synthesis of, harmful information or design capabilities related to chemical, biological, radiological, and nuclear weapons and other dangerous materials	Model (principal); Access Control (secondary). The model gate checks capability evaluation and refusal behavior; the access control gate checks who can reach sensitive capabilities
2	Confabulation	Confidently given false or fabricated content, that is, what is commonly called hallucination or fabrication; users may be misled	Knowledge (principal); Model, Observability (secondary). The knowledge gate governs the basis in the corpus, the model gate governs pre-release evaluation, and the observability gate governs the monitoring of live outputs
3	Dangerous, Violent, or Hateful Content	Easier generation of, and exposure to, violent, inciting, and threatening content, and advice on self-harm or illegal activity; hateful and demeaning content is difficult to control	Gateway (principal); Model (secondary). The point where content enters and leaves the system is the gateway gate; the model gate checks safety evaluation
4	Data Privacy	Impacts arising where personal or sensitive data, such as biometrics, health, and location, are leaked, used without authorization, disclosed, or de-anonymized	Memory (principal); Access Control (secondary). What may be stored, and for how long, is at the memory gate; who may access it is at the access control gate
5	Environmental Impacts	The high computing consumption of training or running generative AI models, and its adverse effects on the ecology	No coverage. The cost monitoring of the observability gate covers only computing expenditure, which is not an environmental impact evaluation; this item corresponds to MEASURE 2.12 of the RMF and lies outside the scope of the ten gates
6	Harmful Bias or Homogenization	Historical, social, and systemic bias is amplified; performance differs across subgroups or languages; outputs converge toward sameness and mislead decisions	Model (principal); Knowledge (secondary). Note that the ten gates do not name a fairness check; this item depends on extending the evaluations of the model gate. The knowledge gate governs the representativeness of the corpus

#	Risk (original AI 600-1 name)	Main Points of the Definition	The Gate That Checks It
7	Human-AI Configuration	The configuration of, and the interaction between, humans and the AI system may lead to anthropomorphism, algorithm aversion, automation bias, over-reliance, or emotional entanglement	Alignment & Autonomy (principal); Access Control (secondary). The degree of autonomy and the escalation paths are at the alignment and autonomy gate; human approval of key actions is at the access control gate
8	Information Integrity	The threshold falls for generating content that does not distinguish fact from opinion or fabrication and does not mark uncertainty, and such content can be used for large-scale disinformation activities	Knowledge (principal); Observability (secondary). The knowledge gate governs source quality and traceability, and the observability gate governs the discovery of anomalous patterns; output watermarking and content provenance are not within the ten gates
9	Information Security	The attack threshold falls (automated vulnerability discovery and exploitation, malware, phishing, offensive cyber operations), the attack surface widens, and the confidentiality and integrity of training data, code, and model weights are endangered	Access Control (principal); Tool Use, Gateway (secondary). The access control gate governs access and permissions, the tool use gate governs the injection risks brought by tool outputs, and the gateway gate governs entry filtering
10	Intellectual Property	Easier generation or reproduction of unauthorized content protected by copyright, trademark, or license; easier leakage of trade secrets; or plagiarism and unlawful copying	Knowledge (principal); Model (secondary). The knowledge gate governs corpus sources and authorization; organization-level intellectual property policy belongs to GOVERN 6.1 and lies outside the ten gates
11	Obscene, Degrading, and/or Abusive Content	Easier generation of, and exposure to, obscene, degrading, and abusive imagery, including synthetic child sexual abuse material and non-consensual intimate imagery	Gateway (principal); Model (secondary). The same inspection path as item 3: gateway filtering plus model safety evaluation
12	Value Chain and Component Integration	Upstream third-party components (including improperly obtained data) are integrated without transparency or traceability, and vendor review is inadequate, harming transparency and accountability toward downstream users	Tool Use (principal); Platform (secondary). This is the item among the 12 that the ten gates cover most completely: the tool use gate governs component admission and review, and the platform gate governs integration and deployment

D.5 Positioning Statement

DeepWing's seven layers and ten gates are an inspection method for AI systems that are already running in production. This crosswalk maps each of the ten gates to the functions and subcategories of the NIST Artificial Intelligence Risk Management Framework 1.0 (NIST AI 100-1), with the Generative Artificial Intelligence Profile (NIST AI 600-1) as a supplementary risk lens. The AI RMF operates at the governance level: what an organization should consider, who is accountable, and how decisions are made. The ten gates operate at the system level: where, inside a running enterprise AI system, an inspector should look, and what evidence to ask for. Aligned in this way, DeepWing's tools serve as the operational TEVV checklist for production AI systems, translating governance questions into inspectable checkpoints. This document is a reference crosswalk only. It is not a NIST certification, it implies no endorsement by NIST, and it states no compliance conclusion. The mapping was produced independently by DeepWing and has not been reviewed or approved by NIST.

End Matter I: Gap List (Unresolved Matters, Formal Edition v1.0)

The following matters are those that the source materials and the ratified documents cannot support, and that this edition, as a matter of discipline, leaves blank or addresses only by method. They will be supplemented in later versions after a revision is recorded under Annex C.

1. Self-check question text to be transcribed back: the original question text of the 17 Quick Self-Check items exists only in the current version of the self-check instrument, and the ratified documents retain only the item numbers and gate themes. The check items in each layer of this document and the gates in Chapter 12 are all matched by item number, and the question text is to be transcribed back from the current version of the self-check instrument and then supplemented into a supplementary table to Chapter 13.
2. Quantitative thresholds missing: routing tier proportions, degradation trigger conditions, monetary and scope thresholds for confirmation points, alert thresholds, and scale ceiling figures. At this stage only principles and individual case figures are available, which cannot be raised to normative thresholds. This edition states only the determination methods (see the note to 6.5.4, 11.3.4, and elsewhere), and the figures are left for later versions to supplement on the basis of accumulated cases and measurements.
3. Method for checking silent vendor changes pending supplementation: the MOD register already has registration practice, but the method clauses on "how an unannounced version change is discovered and how it is handled once discovered" have not been written. Clause 5.5.5 establishes only the registration and re-review requirements.
4. Normative references pending item-by-item verification: for the external documents in 2.3 of Chapter 2, this edition sets no compliance obligation; the existing reference materials follow a further reading format, and the version and date of each item shall be verified and its actual citation by a clause confirmed before it may be raised into the list of normative references.
5. Model cases pending attachment: there is only one case whose primary layer is Model, INC-023 (with the secondary-layer association INC-022); the five failure modes in 5.4.1 to 5.4.5 have no case available to attach yet (see 5.8.3).
6. Memory cases pending attachment: 8.4.3 cross-user bleed and 8.4.4 not forgetting what should be forgotten have no case available to attach yet (see 8.8.3).
7. Observability cases pending attachment: 11.4.5 alert fatigue and 11.4.6 disclosure lag have no case available to attach yet; the latter awaits completion of the backfilling of the disclosure lag field in the case library before supplementation (see 11.8.4).

8. Governance readings pending construction: the maturity readings of DW-AI-GOV can at present give only provisional readings for a small number of cells (for the state of its data, see Section V of DW-AI-GOV v1.0); clauses 12.1.5, 12.7.5, 12.8.5, 12.9.5, 12.10.5, and 14.4.5 of this document establish only the interface conventions and do not prefill reading conclusions.
9. Source materials vary in depth across the gate clauses: Gate 9, Platform, and Gate 10, Alignment & Autonomy, in Chapter 12 rest mainly on principled discussion and on the conventions of the Ratified Version, and their case support is weaker than that of the other gates. Later versions should supplement the detailed determination rules with new cases and registration practice.

End Matter II: Conventions (Applied Throughout)

I. Numbering Rules

1. Chapters are numbered with integers (Chapter 1, Chapter 2, and so on). The layer chapters are arranged in the order of the seven layers: Model, Orchestration, Tool Use, Memory, Knowledge, Access Control, Observability, as Chapters 5 to 11 respectively; gates do not form separate chapters and are uniformly included in Chapter 12, numbered 12.1 to 12.10 in check order.
2. Clauses are numbered as "chapter number.sequence number" (for example, 5.1, 5.2), and where a clause is further divided into sub-clauses, as "chapter number.clause number.sub-clause number" (for example, 5.5.5). Clause citations shall always give the full number, and relative references such as "the preceding clause" or "Section 3 of this chapter" shall not be used.
3. The check-item number format is "full layer name-C", for example Model-C1, Orchestration-C3, Access Control-C10: the layer full names are Model, Orchestration, Tool Use, Memory, Knowledge, Access Control, and Observability. The numbers are used only for check-item numbering and cross-references, and narrative text uniformly uses the full official name. The two-letter code format used previously is discontinued from this edition.
4. Cases and register entries state IDs only (for example, INC-023, MOD-012) and not link addresses; where the same event crosses registers, it is numbered separately in each register, with cross-listed links.

II. Terminology Conventions

1. On first occurrence, a term is written in the form "official name (English name)", and thereafter only the official name is used. The English names of the seven layers and the ten gates follow the Ratified Version: the seven layers are Model, Orchestration, Tool Use, Memory, Knowledge, Access Control, Observability; the ten gates in English are Gateway, Access Control, Orchestration, Knowledge, Memory, Tool Use, Model, Observability, Platform, Alignment & Autonomy.
2. Retired names in the Ratified Version shall not be used in the body text, including: Gating, Oversight, Knowledge Base, Model Layer, AI Gateway, Enterprise AI Platform. Where a retired name appears when historical materials are cited, the current official name shall be annotated.

3. The definition paragraph in the first section of each layer chapter is the source of definition for that layer's terms. Where another chapter cites them, the definition is not repeated and only the defining clause number is given.

III. Clause Conventions

1. Clause wording is graded: "shall" expresses a requirement, "shall not" expresses a prohibition, "should" expresses a recommendation, and "may" expresses a permission. Pass determinations for check items are made only for clauses of the "shall" and "shall not" kinds.
2. The fixed small structure of each layer chapter is: Scope and Definitions / Responsibility Boundaries / Constituent Elements / Common Failure Modes / Normative Clauses / Check Items / Interfaces with Other Layers and Gates / Related Cases. The eight sections are neither added to nor subtracted from, and their order is not changed.
3. Every sentence shall belong to one of the three attributes: a source fact, a DeepWing interpretation, or unknown. A factual statement in the clauses shall be able to point to its source materials; an interpretation is marked uniformly at the head of the chapter or the head of the section and is not interspersed with the clauses so as to pass as a fact.
4. The body text does not use long dashes (em dash and en dash). Pauses and parallel items use commas, enumeration commas, or separate lines; placeholders uniformly use "·".

IV. Version Record Format

1. The version number format is v[major version].[minor version] (for example, v1.0, v1.1): a structural adjustment (adding or removing a chapter, or changing layer or gate conventions) raises the major version, and a clause supplement or a text revision raises the minor version.
2. Each line of the revision record in Annex C has six fields: date, version, chapters revised, revision content, basis (a case ID, a register ID, or a ratified document), and revised by.
3. The status words for entries and records are uniform: In Force, Corrected, Withdrawn. For a Corrected entry, the old statement remains in the change record; for a Withdrawn entry, the ID is never reused.
4. Every record retains a "Last Verified" date; the re-verification cycle for framework clauses follows Chapter 14, and where a clause is overturned by a case, the change record is entered first and the body text is corrected afterward.