Charterion™

Agentic Enterprise Architecture Framework

Version 1.1 — September 2026

A formal, continuous and verifiable enterprise architecture framework for organisations acted upon by autonomous software agents

The metamodel, the semantics, the method, the governance model, the artefacts, the conformance levels and the worked cases of an enterprise architecture in which agents are principals, authority is derived from business scope, and every answer carries its warrant.

Created by Samir El Hassani
Independent researcher, Accenture Strategy & Consulting, Paris, France
ORCID 0009-0005-8880-3872 · samirelhassani1998@gmail.com

Charterion™ — Agentic Enterprise Architecture Framework
Version 1.1 — September 2026
Created by Samir El Hassani

Legal notice

The views expressed are the author's own and do not represent those of Accenture or its clients. No client or product data was used. The framework is provided *as is*; it is a research artefact whose validation status is stated in Chapter 26, and nothing in it constitutes legal, regulatory or security advice.

Citation. El Hassani S (2026) Charterion™ — Agentic Enterprise Architecture Framework, Version 1.1. Paris, September 2026.
Version history. Version 1.0 (September 2026): first complete release, consolidating the author's five research articles of 2026 and adding the constructs marked *New in Charterion v1.0* throughout.
Name. The framework was developed under the working name *AEAF* in the author's 2026 articles and in the first revision of this document; the acronym is already used in the literature for an *Agile Enterprise Architecture Framework*, and an *Enterprise Agentic Architecture Framework* (EAAF) has been published with a near-identical title. It was renamed *Charterion* in September 2026—from the *charter*, the construct by which every authority in the framework is conferred—so that the mark is distinctive and can be claimed and protected. References to AEAF in the articles designate this framework. The version policy is stated in Appendix H. Version 1.1 (September 2026): minor version under the policy of Appendix H; adds eight optional profiles (**WF13**–**WF19**, **CC7**) in a new stratum, authority certificates, Table 24.2-bis re-coded from primary texts, computed regulatory evidence, a controlled injection study and the Appendix H regression; corrects errata E1–E16; changes no semantics (Appendix Q).

Preface

Enterprise architecture has been written, for four decades, for a human reader. Its frameworks reconcile the perspectives of stakeholders, its viewpoints serve the informational needs of distinct audiences, and its purpose has been described as sensemaking. Its quality theory measures comprehension by people, and its coverage doctrine—model the essentials and no more—is affordable only because a human reader interpolates what the model omits.

Agentic artificial intelligence changes the reader. Systems built on large language models now plan, call tools and act on the records of the organisation; they hold authority that somebody conferred on them, they delegate to further agents, and they do so without a person at every step. An enterprise model that such a system consults must answer questions the classical frameworks were never asked: *Does the thing I intend to act on exist, and is it the one I mean? Can it be done, through what, and with what consequence? May I do it, and where must I stop? With whom am I sharing what I am about to change? How far can I trust the answers I have just been given?* And the architecture function that governs such a system must answer questions of its own: *Which human is accountable for this agent's write access? Which two agents may act on the same record without coordination? Which of this agent's entitlements exceed any need the architecture can justify?*

Charterion™ (Agentic Enterprise Architecture Framework) is an answer to both sets of questions. It was developed through a design-science research programme conducted in 2026 and reported in five articles: an argumentative research note that stated the thesis of enterprise architecture as *execution substrate*; a survey of 150 practitioners that confronted the thesis with the profession and located its largest gap in the constraint envelope; the Substrate Framework, which gave the run-time substrate a metamodel, a four-valued admissibility function, a lifecycle, invariants and conformance levels; the Boundary Substrate, which extended it to agent-to-agent business across organisations; and *Agency as an Architecture Layer*, which added agency to the enterprise metamodel itself and gave it a stratified-Datalog semantics with proven properties. Each article was written to stand alone. This document does the opposite: it assembles the five into one framework, states in one place its metamodel, its semantics, its method, its governance model, its artefacts and its conformance levels, and adds what a framework needs and a research article does not—the coherence conditions that bind design time to run time, the governance mechanisms expressed as checkable model conditions, the viewpoints and catalogues an architecture team will actually keep, an exchange format, worked cases in five sectors, and an honest statement of what has and has not been validated.

Three commitments shaped the writing. The first is *formality where it pays*: every construct that an analysis depends on is defined so that a program can evaluate it, and the properties the framework relies on are proved, not asserted. The second is *continuity with the discipline*: Charterion keeps the layered enterprise model of ArchiMate, maps its method onto the phases of the TOGAF ADM, and is designed to be adopted as an extension—an ArchiMate specialisation profile and a TOGAF-compatible method—rather than as a replacement that discards the investment organisations have made. The third is *candour about novelty and validation*. Much of what agents need—identity, sponsors, scoped and attenuating delegation, autonomy levels, runtime enforcement—has been articulated since 2025 by security and standards bodies, and this document cites that work and positions itself against it. What Charterion adds is not that agents need authority, but that authority belongs in the enterprise model, can be derived from business scope through the model's own structure, can be checked at design time by a tractable procedure, and can be bound to run-time and cross-organisational enforcement by conditions

that a repository can verify on every change. That claim is stated precisely in Chapter 24, and its evidence—analytical, experimental and demonstrative, not yet naturalistic—is stated with equal precision in Chapter 26.

New in Charterion v1.1

Version 1.1 (September 2026). This minor version keeps every definition, condition and algorithm of Version 1.0 and adds what the reviews of 1.0 asked for: constructs with no counterpart elsewhere (configuration currency, chartered instantiation, staffed reservation, bounded cap composition, grounded obligation, delegation perimeter, bounded blast radius, authority certificates), machine-checkable rigour (a general preservation theorem for the new stratum, a controlled injection study, a mechanical regression against 1.0), standards bindings (the OWASP Agent Control Standard, the IETF AIMS draft, XACML, AuthZEN, the MCP servers of three EA tools) and evidence stated by kind. The landscape of Chapter 24 was re-read from primary texts under a written protocol; where 1.0 had claimed too much, 1.1 says so. Whether the change of premise is a revolution of the discipline remains, as 1.0 said, a matter of validation and adoption.

Samir El Hassani
Paris, September 2026

How to read this document

Structure

The document has ten parts and appendices. **Part I** states the problem, the principles and the overall architecture of the framework (three planes, one grid). **Part II** is the content metamodel with its formal semantics: the core enterprise model, the agency layer, the execution-plane substrate, the boundary plane, the coherence conditions that bind them and the optional extension profiles. **Part III** is the method: a continuous cycle with a design loop, an operation loop, a boundary loop and the bridges between them. **Part IV** is the governance model: roles, bodies, decision rights, the charter lifecycle, invariants as gates, regulatory alignment and metrics. **Part V** catalogues the artefacts and viewpoints; **Part VI** describes the reference implementation and the exchange format. **Part VII** works five sector cases through the framework with computed results. **Part VIII** positions Charterion against classical EA frameworks and against the agentic-AI governance and security work of 2025–2026. **Part IX** defines the maturity model and conformance statements. **Part X** states the validation status, the testable propositions and the limitations. The appendices consolidate the formal definitions, the glossary and notation, the requirements traceability, the ArchiMate profile and TOGAF mapping, the exchange-format schema and the version policy.

What this document claims, and what it does not

Charterion does not claim to have invented the agent as a principal, the human sponsor, scoped or attenuating delegation, autonomy levels, human oversight or run-time interception; these are stated, since 2025, by the Cloud Security Alliance, NIST, OWASP, IMDA, Google and others, and Chapter 24 credits them. It claims a change of premise—the enterprise model as an authority substrate consumed by machines rather than a description consumed by people (Chapter 3)—and the constructs that follow from it: authority derived from business scope, a formal semantics for agent authority with proofs, coherence between design, run time and boundary, and, as secondary contributions, the conditions that meet failures specific to agentic enterprises (**WF8**–**WF12**, the Authority Request Protocol). The five central contributions are stated in Section 24.2; the protocol by which the framework is to be evaluated independently is in Section 26.5. The author's position is that Charterion is a *candidate* next-generation EA paradigm: whether the change of premise is a revolution of the discipline is not for a specification to assert; it is a matter of validation and adoption, and Chapter 26 states exactly how far each has gone.

New in Charterion v1.1

Version 1.1. Having re-read the landscape from primary texts, this version also does not claim run-time step-up through a human, agent bills of materials, decision outcomes beyond permit and deny, holder-attenuable delegation tokens or machine access to EA repositories, all of which exist; Section 24.5 names who provides them. It claims, in addition to the constructs of 1.0, an independently checkable derivation object for every entitlement (Section 11.5), and it narrows its per-assertion warrant claim to the four elements no retrieved work matches. The evidence for every v1.1 construct is analytical, experimental on synthetic models, or demonstrative on the author's cases; none is naturalistic (Table 26.2).

Normative language

This document uses the conventions of standards writing. SHALL and SHALL NOT state requirements that a Charterion-conformant model, method application or tool must meet; SHOULD states recommendations whose neglect must be justified in the Conformance Statement; MAY states permissions. Text not marked in this way is informative. Numbered *Definitions*, *Conditions* and *Propositions* are normative; *Rationale* and *Remark* paragraphs and all boxed *notes* are informative.

Provenance marks

Constructs inherited unchanged from the author's research articles are cited to them at first use. Constructs introduced by this document are marked with a box headed *New in Charterion v1.0*, so that a reader of the articles can see at once what the framework adds, and so that the validation status of each construct (Chapter 26) can be read against its provenance. Additions of Version 1.1 are marked with a box **New in Charterion v1.1** and are optional: no v1.1 condition is required at any conformance level (Appendix H). The change log is Appendix Q.

Audiences

Enterprise and solution architects will find the metamodel (Part II), the method (Part III) and the artefacts (Part V) sufficient to model an organisation's agents and run the analysis; Part VII shows what the results look like. *Risk, security and compliance functions* will find the governance model (Part IV) and the regulatory alignment tables their entry point. *Agent platform teams* will find in Part VI what they must consume (derived needs, envelopes) and produce (decision logs), and in Part VIII how Charterion relates to the runtime and identity frameworks they already use. *Researchers* will find the formal definitions in Part II and Appendix A, the design-science provenance in Part I, and the testable propositions in Part X.

Contents

List of Tables

List of Figures

Part I

Foundations

1 Purpose, scope and conformance

1.1 Purpose

Charterion is an enterprise architecture (EA) framework for organisations in which software agents—systems built on large language models or comparable planners that select and call tools and act on business systems—perform work under authority conferred by the organisation. Its purpose is threefold.

1. To give such an organisation an *enterprise model in which agents are principals*: elements of the architecture on a par with people, applications and technology, whose authority is stated at the level of business scope, derived through the model to the resources they touch, traceable to accountable humans and analysable before deployment.
2. To give the agents themselves an *execution substrate*: a machine-readable, warrant-bearing description of the organisation that a program consults at the moment of acting and that returns one of four answers—permitted, forbidden, permitted subject to a human decision, or not warranted—together with the evidence for that answer.
3. To bind the two by *coherence conditions* that a repository can check on every change, so that the authority an agent exercises at run time, at home or across an organisational boundary, is never more than the authority the architecture has justified, and so that the evidence produced at run time returns to the architecture that caused it.

1.2 Scope

Charterion covers: the content metamodel (core enterprise model, agency layer, substrate, boundary), its formal semantics and the properties proved of it; a continuous method with a design loop, an operation loop and a boundary loop; a governance model (roles, bodies, decision rights, lifecycle, invariants, metrics, regulatory alignment); the artefacts, viewpoints and exchange format; a reference implementation; conformance levels and a maturity model; and worked cases. It applies to any organisation that intends to let agents read, propose, commit or autonomously change the state of its systems, whatever the agent technology, and to any pair of organisations whose agents transact with one another.

Charterion does not cover: the internal design of agents (planning, memory, prompting); model-level safety and alignment; threat modelling and penetration testing of agentic systems; the cryptographic mechanics of identity, credentials and attestation; or the interception of tool calls at run time. For each of these it names, in Chapter 24, the bodies of work it relies on and the interface through which it relates to them. Charterion is the layer that tells those mechanisms *what* to enforce and *why*; it is not itself an enforcement mechanism.

1.3 Conformance targets

Three things can conform to Charterion.

Model conformance

An enterprise model conforms to Charterion at a stated conformance level (Chapter 25) if it can be expressed in the Charterion Model Exchange Format (Section 21.6), if the well-formedness conditions required at that level hold or every failing condition is recorded in

the Accepted-Defect Register with a rationale, and if the coherence conditions required at that level hold.

Method conformance
: An application of the method conforms if the phases of Part III are performed with the artefacts named for them, if the analysis is executed by a tool that implements the semantics of Part II, and if the governance gates of Part IV are applied at the transitions of the charter lifecycle.

Tool conformance
: A tool conforms if, on the test models supplied with the reference implementation, it derives the same least model and reports the same defect sets as the reference reasoner, and if it reads and writes the exchange format.

A Conformance Statement (Section 25.4) records which target is claimed, at which level, for which slice of the organisation, and which recommendations were not followed and why.

1.4 What Charterion is not

Charterion is not a replacement for an organisation's existing EA framework. It is designed to be adopted as an extension of ArchiMate (through the specialisation profile of Appendix E) and to coexist with the TOGAF Architecture Development Method (through the mapping of Appendix F). It is not an identity-and-access-management product, a policy engine or an agent runtime; it produces the target entitlement set such products should enforce and consumes the decision logs they produce. It is not a code generator: agents are not generated from the model, they consult it. And it is not a governance overlay of prose controls: every control it states is a condition on a model that a program evaluates.

2 The agentic-driven enterprise

2.1 Definitions

Definition 2.1 (Agent)**.** An *agent* is a software system that receives a goal, decomposes it into steps, selects and invokes operations on the organisation's resources, observes the results and iterates, and that does so under authority conferred on it by the organisation rather than under the direct control of a person for each action. An agent is a *non-human principal*: it has an identity to which authority and accountability attach, it may be instantiated many times, and its behaviour is bounded by the operations it can reach rather than by a fixed programme (Yao et al., 2023; Schick et al., 2023; Wang et al., 2024; Xi et al., 2025).

Definition 2.2 (Agentic-driven enterprise)**.** An *agentic-driven enterprise* is an organisation in which agents are principals of its business processes: at least one process step designated by the business is performed by an agent at a level at which the agent changes the state of a resource of the organisation, and the organisation intends the population of such agents, and their delegations to one another, to grow.

The definition is deliberately about *authority*, not about technology. An organisation that uses language models to draft text for people to approve is not agentic-driven in this sense; one in which a settlement agent pays claims, a provisioning agent grants access or a procurement agent places orders is, whatever the model behind it.

2.2 What changes

Four things change when agents become principals, and each defeats an assumption on which classical EA frameworks rest.

The reader of the model changes. The enterprise model acquires a machine reader that cannot interpolate. A human architect who finds no element for a payment queue assumes one exists; an agent that finds none either acts on the wrong object or does not act. The model must therefore say what it does not know (El Hassani, 2026a).

The holder of authority changes. For the first time, an active-structure element other than a person holds authority to change the state of the organisation. That authority is *derived*: conferred by a human, bounded by a scope, and possibly passed on to other agents. None of the classical frameworks has an element for it, because until recently nothing but people held such authority (El Hassani, 2026e).

The tempo of change changes. An agent population changes with every deployment, every prompt revision and every new tool; entitlements administered by hand cannot follow, and an architecture reviewed by projects cannot govern what changes daily. Governance must become a computation the repository performs on every change.

The perimeter changes. Agents transact with the agents of other organisations. The counterparty's agent presents an authority its own organisation conferred; the receiving organisation cannot approve it in its own sense and must weigh it by a trust it controls. Authority becomes portable and warrant becomes asymmetric (El Hassani, 2026d).

2.3 The eight questions

The framework is organised around two sets of questions, one asked *by* an agent at the moment of acting and one asked *about* agents by the architect at design time.

The five questions of an actor (execution plane)

Q1. *Anchor* — Does the thing I intend to act on exist, and is it the one I mean?
Q2. *Affordance* — Can the action be performed on it, through what, and with what consequence?
Q3. *Envelope* — May I perform it, and where must I stop?
Q4. *Coordination* — With whom or what am I sharing the estate I am about to change?
Q5. *Warrant* — How far can I trust the answers I have just been given?

The three questions of an architect (design plane)

A1. Which human is accountable for this agent's write access?
A2. Which two agents may act on the same record without any declared coordination?
A3. Which of this agent's entitlements exceed any need that the architecture can justify?

The survey of 150 practitioners that preceded the framework (El Hassani, 2026b) found that the importance practitioners attach to the first four actor questions exceeds the capability of their current models on every one of them, with rank-biserial effect sizes between .85 and .93 and the largest gap on the envelope question; that 89 % demand per-assertion provenance and confidence; that 83 % accept agent-maintained models only under human validation of authorisation changes; and that 77 % prefer oversight bounded by the model itself over per-action approval or ex-post audit. The three architect questions were shown in the fifth article to be unanswerable from the entitlement and catalogue views organisations currently maintain (El Hassani, 2026e).

2.4 Ten gaps of the classical frameworks

Table 2.1 states the gaps that Charterion closes, with the framework element that closes each. The gaps are stated against the Zachman Framework (Zachman, 1987), the TOGAF Standard, 10th Edition (The Open Group, 2022b), ArchiMate 3.2 (The Open Group, 2022a), and, where relevant, DoDAF (U.S. Department of Defense, 2010) and FEAF (Office of Management and Budget, 2013). They are gaps *for an agentic-driven enterprise*; the frameworks were not designed for one and are not criticised for serving the purpose they were designed for.

Rationale. The gaps are not oversights of the frameworks' authors. Each follows from a premise that was true when the frameworks were written: that the only principals are people and the only automation is application logic; that the model is read by people; that the organisation changes at the pace of projects; and that the organisation's perimeter is crossed by people and documents. Agentic AI falsifies each premise, and the gaps follow.

Table 2.1. Gaps of the classical EA frameworks for an agentic-driven enterprise and the Charterion element that closes each

Id	Gap	Consequence for an agentic-driven enterprise	Closed by
G1	No element for an autonomous software agent as a *principal*: active structure is people (actors, roles) or application components; DoDAF's Performer admits automated performers but carries no authority, scope or accountability.	Agents are modelled as an application component or a performer, and their derived, bounded, delegable authority and their accountable human have no place in the model.	Agent, charter, delegation (Chapter 7)
G2	Authority is not an architecture concern: permissions live in IAM, disconnected from business scope.	Entitlements cannot be justified from the architecture; over-provisioning and shadow grants are invisible.	Need derivation, **WF3** (Chapter 7)
G3	No formal semantics: models are diagrams read by people, not knowledge bases computed over.	Consistency of the agent population cannot be checked; conflicts appear in production.	Stratified-Datalog semantics, **WF1–WF12** (Chapters 7 and 8)
G4	The method is a project cycle (ADM phases, transition architectures).	An agent population that changes daily cannot be governed by a cycle measured in months.	Continuous method, repository checks (Part III)
G5	The model is addressed to a human reader who interpolates omissions.	A machine reader acts on the wrong object or does not act; the model cannot say what it does not know.	Warrant-bearing assertions, *unwarranted* outcome (Chapter 9)
G6	No run-time role: the architecture describes, it does not decide.	Agents are grounded on documents and tool descriptions, not on the architecture.	Admissibility function, execution plane (Chapter 9)
G7	Human oversight is a principle, not a construct.	Autonomy cannot be checked against oversight; regulation (Article 14 of the AI Act) cannot be shown to be met from the model.	Oversight, escalation, **WF7**, **WF9** (Chapters 7 and 8)
G8	Separation of duty and coordination are workflow or security concerns, not architecture concerns.	Two agents that both write the same record from concurrent steps are discovered in production.	Guards, **WF4**, **WF5** (Chapter 7)
G9	The enterprise stops at its boundary; inter-organisational architecture is about interfaces and contracts between people.	An agent's authority abroad and the trust placed in a foreign agent's claims are unmodelled.	Boundary plane (Chapter 10)
G10	Governance is by review board and document, evidence returns by audit.	Run-time evidence never reaches the model; charters that are too broad or too narrow are never corrected.	Coherence conditions, evidence-gated promotion (Chapters 11 and 18)

3 The paradigm shift: from description to authority substrate

3.1 What changes is the addressee

Every classical EA framework, whatever its grid or cycle, shares one premise: the model is written for a human reader who will decide something and then act through projects and people. Its quality criteria (comprehensibility), its coverage doctrine (model the essentials), its maintenance economics (decay is tolerable because the reader interpolates), its governance (boards and reviews, after the fact) and its languages (structural description of an as-is and a to-be estate) all follow from that premise. Charterion starts from a different premise, stated in the author's research note (El Hassani, 2026a) and confronted with 150 practitioners (El Hassani, 2026b): in an agentic-driven enterprise the model's first reader is a program that acts, and the program cannot interpolate. The consequences are not an extension of the classical arrangement but its inversion on five points (Table 3.1).

Table 3.1. The five inversions implied by the shift of addressee (El Hassani, 2026a), and where Charterion realises each

	Human addressee (established)	Machine actor (inverted)	Realised in Charterion by
Quality	Pragmatic quality binding; notation engineered for cognition	Ontological and logical quality binding; notation irrelevant	Sorted constants, stratified semantics, least model (Chapter 7); UFO grounding (Section 8.8)
Coverage	Purposeful abstraction; model the essentials	Instance-level currency for the slice being acted upon, with an explicit boundary	Slice discipline (Section 12.3); anchor layer and validity τ (Chapter 9); *unwarranted* outcome
Maintenance	Decay tolerable; the human reader interpolates	Decay actionable; the agents maintain the model they are governed by, under conditions	Operation loop O1–O6; **Inv-1**–**Inv-3**; Level 4 (Chapter 9)
Governance	Ex-post conformance via boards and reviews	Run-time authorisation; the model is the policy decision point	Envelope and ADM (Chapter 9); lifecycle gates (Chapter 18); **CC1**–**CC6**
Language	Structural description of an as-is and a to-be estate	Preconditions, effects, authority scope, provenance, uncertainty, intent	Agency layer (Chapter 7); assertion tuple; affordance and coordination layers

3.2 The two pipelines

Figure 3.1 puts the two arrangements side by side. In the established one the model informs a decision and a person closes the loop to the enterprise; staleness and omission are absorbed by the reader. In the Charterion arrangement the model is queried at the moment of acting, returns an admissibility decision with its warrant, the agent acts or escalates, and the evidence of the action returns to the model that authorised it. The architect does not disappear: the architect becomes the author of the authority the agents exercise, and the approver of the assertions the

agents are governed by. What disappears is the interval between architecture and action.

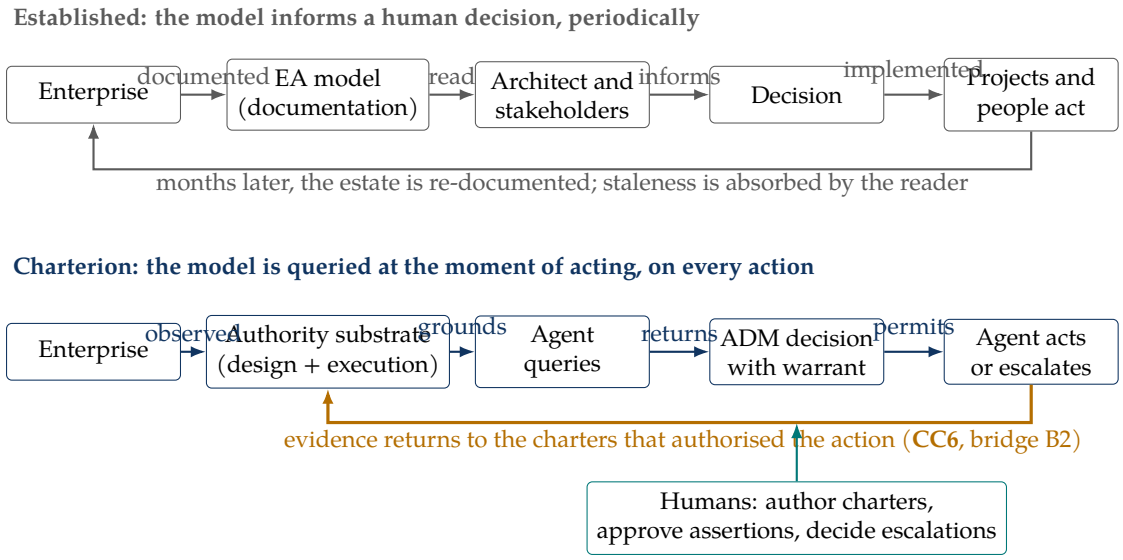

Figure 3.1. The established pipeline and the Charterion pipeline. The difference is not an added layer but the addressee of the model and the length of the loop from architecture to action.

3.3 Why this is not "TOGAF plus agents"

Three things follow from the inversion that no extension of a descriptive framework can supply. First, the model must be able to *refuse*: an assertion the organisation has not approved confers no authority, and an answer the model cannot warrant is returned as *unwarranted* rather than guessed—which requires provenance, confidence and approval status on every assertion, not on the document. Second, authority must be *derived rather than declared*: the entitlement an agent holds on a system is a consequence of the business scope it was chartered for, so that an architect who narrows a charter narrows the entitlements, and an architect who reads the model reads the authority actually in force. Third, the human position must be a *construct*, not a principle: reserved steps, oversight, escalation reachability and accountability chains are facts the semantics reasons over, so that the question "who answers for this action" has one computable answer at every instant. Charterion is compatible with TOGAF and ArchiMate—it can be adopted as a profile and as a method mapped onto the ADM (Chapter 17, Appendix F)—but compatibility is a property of its *artefacts*, chosen for adoption; its *semantics* is what the classical frameworks do not have and cannot acquire by adding elements.

New in Charterion v1.0

Three constructs of v1.0 are specific to this inversion. Each has cousins—workflow satisfiability under delegation, compensating transactions, access-request workflows—and none is claimed as a new idea in general; what has, to the author's knowledge, no counterpart in any EA or agentic-governance framework is their statement as conditions on the enterprise model: *independent separation* (**WF11**), which detects separation-of-duty pairs whose two agents are controlled by one orchestrator; *compensable autonomy* (**WF12**), which admits autonomous authority on irreversible operations only where the model contains a compensation; and the *Authority Request Protocol* (Chapter 16), by which an agent that has been refused may ask the architecture for authority—and the architecture answers through its human gates, never by itself. Together with **WF8–WF10** and **CC1–CC6** they are what the phrase "authority substrate" adds to "enterprise model".

4 Principles and grounding

4.1 Axioms

Six statements summarise the framework and are used throughout as the shortest expression of its intent. They are informative here and are made normative by the conditions of Part II that operationalise them.

The six axioms of Charterion

X1. No authority without a charter. An agent acts only under a charter that binds it to a business scope at a stated level under a stated principal. (*Operationalised by the semantics of Chapter 7 and **CC1**.*)

X2. No charter without a principal. Every charter that confers a write level names a human, or a role with a current holder, who is accountable. (***WF1**, **CC5**.*)

X3. No entitlement without a need. Every provisioned entitlement is justified by a step the agent covers; the architecture determines what is provisioned. (***WF3**, **CC2**.*)

X4. No autonomy without oversight. An agent that acts without per-action confirmation is watched by a reachable human, and no human watches more agents than the organisation has decided a person can. (***WF7**, **WF9**.*)

X5. No mandate beyond charter. An agent holds no more authority abroad than it holds at home, and no more after delegation than before. (*Propositions of Chapters 7 and 10, **CC3**.*)

X6. No answer without warrant. The substrate returns *unwarranted* rather than a default wherever its evidence is absent, expired, ambiguous or too weak. (***Inv-4**.*)

4.2 Design principles

Twelve principles guided the design and constrain future versions. Each is stated with its rationale and its implication.

Principle 1 (Agency is a layer, not an overlay)**.** Agents, their charters, their delegations and the guards on their interaction are elements of the enterprise model, connected to the business layer by scope and principal and to the technology layer by the comparison of need with grant. *Rationale:* the questions organisations ask about agents cross layers, and only an integrated model can answer them (Jonkers et al., 2006; Tamm et al., 2011). *Implication:* Charterion extends the metamodel rather than adding a register beside it.

Principle 2 (Authority is stated at business scope and derived to resources)**.** A charter binds an agent to a capability, process or step; the resource-level needs follow from the realisation structure of the model. *Rationale:* zero-trust architecture requires every access to be justified (Rose et al., 2020); the enterprise model is the only place where the business justification of an access is recorded. *Implication:* provisioning flows from the model to identity management, never the reverse.

Principle 3 (Delegation attenuates and accountability persists)**.** An agent may pass on part of its authority, never more than it holds; the humans accountable for the delegator remain accountable for the delegatee. *Rationale:* the theory of delegation between agents (Castelfranchi and Falcone, 1998; Norman and Reed, 2010) and the non-amplification results of trust management (Li et al., 2002). *Implication:* the meet operator on levels and the propagation of accountability along chains (Chapter 7).

Principle 4 (Four levels, one vocabulary)**.** All authority is qualified by one of four ordered levels—*read*, *propose*, *commit*, *autonomous*—used identically at design time, at run time and at the boundary. *Rationale:* the levels compress the human–automation scale of Parasuraman et al. (2000) to the distinctions that matter for provisioning; a shared vocabulary lets a charter and a run-time decision speak of the same thing. *Implication:* finer scales (Section 8.9) are mapped onto the four, not substituted for them.

Principle 5 (Formal where analysis depends on it)**.** Every construct the analysis uses has a definition a program can evaluate, and every property the framework relies on is proved. *Rationale:* EA analysis research has converged on the model as a formal knowledge base (LePendu and Dou, 2011; Barbosa et al., 2019); a framework whose conditions cannot be computed cannot be continuous. *Implication:* stratified Datalog over a finite lattice as the semantics; polynomial complexity as a requirement.

Principle 6 (The model says what it does not know)**.** Every assertion carries provenance, confidence, validity and epistemic status, and the substrate answers *unwarranted* rather than guessing. *Rationale:* the machine reader cannot interpolate; 89 % of surveyed practitioners demand per-assertion provenance (El Hassani, 2026b). *Implication:* warrant is a cross-cutting layer of the substrate, and the fourth outcome of admissibility.

Principle 7 (Observed and approved are kept apart)**.** Agents may maintain the model, but content they observe never authorises; only content a human has approved does, and no agent approves. *Rationale:* the condition under which practitioners accept agent-maintained models (El Hassani, 2026b); the self-authorisation hazard (Chan et al., 2024). *Implication:* **Inv-1**–**Inv-3**, and the separation of the design plane (approved by construction) from observation.

Principle 8 (Human reservation is explicit)**.** Steps the business reserves to people are stated as such, and no agent may hold write-level authority over them. *Rationale:* the granularity study of El Hassani (2026e) showed broad charters conferring write authority over hundreds of human steps; regulation requires meaningful oversight of consequential decisions (European Parliament and Council, 2024; IMDA, 2026). *Implication:* the Reserved predicate and **WF8** (new in v1.0).

Principle 9 (Oversight must be reachable and bounded)**.** Every agent with write-level authority has a current human who decides its escalations, and no human's span of oversight exceeds a declared bound. *Rationale:* oversight that names a vacant role, or a person watching a hundred agents, is oversight in name only; automation bias grows with span (Parasuraman et al., 2000; IMDA, 2026). *Implication:* Escalation, oversight span and **WF9** (new in v1.0).

Principle 10 (Authority is finite in time)**.** Charters carry a validity interval and a review date; expired charters confer nothing, and charters past review confer less. *Rationale:* agents outlive the reorganisations that created them; **Inv-4** requires every assertion to have a finite validity. *Implication:* the temporal profile and **WF10** (new in v1.0).

Principle 11 (Design time governs run time, and run time informs design time)**.** The run-time envelope of an agent is derived from its charters, its portable mandate is bounded by them, and the evidence of its actions returns to the charter that caused them. *Rationale:* without the first, the architecture is documentation; without the second, it never learns. *Implication:* the coherence conditions **CC1**–**CC6** and the two bridges of the method (new in v1.0).

Principle 12 (Adopt by extension, slice by slice)**.** Charterion is introduced as an ArchiMate specialisation profile and a TOGAF-compatible method, and applied to one operational slice at a time, each slice at its own conformance level. *Rationale:* the profession will not discard its repositories, and no organisation moves from documentation to a warrant-bearing substrate in one step (El Hassani, 2026c). *Implication:* the profile and mapping appendices and the slice discipline of the method.

4.3 Theoretical grounding

Charterion draws its justificatory knowledge from five bodies of work, each of which supplies one part of the design and none of which supplies the whole.

Enterprise architecture and model analysis supply the layered, cross-layer enterprise model (Lankhorst, 2017; The Open Group, 2022a) and the position that such a model is a knowledge base over which properties are computed (Iacob and Jonkers, 2006; Antunes et al., 2014; Hinkelmann et al., 2016; Barbosa et al., 2019). *Multi-agent organisations* supply the idea that an organisation can be an explicit, machine-readable artefact constraining what agents may do (Zambonelli et al., 2003; Hübner et al., 2007; Esteva et al., 2001; Boissier et al., 2013), the normative constructs of obligation, permission and prohibition (Boella et al., 2006; Vasconcelos et al., 2009), and the theory of bounded delegation (Castelfranchi and Falcone, 1998; Norman and Reed, 2010). *Access control and logic-based policy* supply role- and attribute-based models (Sandhu et al., 1996; Ferraiolo et al., 2001; Hu et al., 2014), zero trust (Rose et al., 2020), Datalog-based trust management with its decidable, tractable delegation (Li et al., 2002, 2012), workflow authorisation constraints (Bertino et al., 1999; Crampton, 2005) and the theory of enforceable policies (Schneider, 2000); Datalog itself supplies the least-model semantics and polynomial data complexity (Ceri et al., 1989; Abiteboul et al., 1995; Dantsin et al., 2001). *Human–automation interaction and AI governance* supply the ordering of levels of automation (Parasuraman et al., 2000), the case for high automation with high human control (Shneiderman, 2020), the analysis of agentic risk (Chan et al., 2023, 2024; Gabriel et al., 2024; Hammond et al., 2025) and the legal requirement of human oversight (European Parliament and Council, 2024). *Inter-organisational systems and commitments* supply social commitments and their lifecycle (Singh, 1999; Telang and Singh, 2012), institution-based trust (Zaheer et al., 1998; Pavlou, 2002) and the attenuation principle of capability-based protection (Dennis and Van Horn, 1966).

4.4 Empirical grounding

The framework's requirements are grounded in the survey of 150 practitioners (El Hassani, 2026b) summarised in Section 2.3; its constructs were exercised on scenarios and synthetic models whose results are reported in Chapter 26; and the fifth article's projection study established, for the design plane, which conditions become undetectable when the enterprise model is reduced to the views current practice maintains (El Hassani, 2026e). What the framework has not yet had is a naturalistic evaluation on an organisation's own repository; Chapter 26 states this and Section 26.6 sets out how it is to be obtained.

4.5 Design-science provenance

Charterion was produced by a design-science research programme (Hevner et al., 2004; Peffers et al., 2007) and constitutes, in the terms of Gregor and Hevner (2013), a nascent design theory: constructs, a model of their relations, a method, instantiations and testable propositions. Table 4.1 traces the DSR activities to the articles and to the parts of this document.

Table 4.1. Design-science activities and their realisation in the Charterion programme

Activity	Realisation	Where
Problem identification	Displacement of the human reader of the enterprise model by a machine actor; agents as principals without a place in the metamodel	Research note; Chapter 2
Objectives	**DR1**–**DR6** (design plane), **SR1**–**SR12** (execution plane), **XR1**–**XR6** (boundary), grounded in the survey and the literature	Section 5.4; Appendix D
Design and development	Metamodel, semantics, method, governance, artefacts, conformance levels; v1.0 additions	Parts II–V
Demonstration	Reference implementation on five sector cases and a boundary scenario	Part VII
Evaluation	Proofs; scalability, projection and granularity studies; simulation of warrant; requirements traceability; comparative analysis	Chapter 26; Chapter 24
Communication	Five articles (2026); this specification	—

5 Architecture of the framework

5.1 Three planes

Charterion organises its content into three planes, each of which answers one question, and binds them by coherence conditions (Figure 5.1).

The design plane
: is the enterprise model with its agency layer. It answers the architect's question: *is the architecture well formed with respect to the agents it hosts?* Its content is approved by construction—it is the organisation's statement of what its agents may do—and its analysis is a computation over the model that runs on every change (Chapters 6, 7 and 8).

The execution plane
: is the substrate an agent queries at the moment of acting. It answers the actor's question: *is this request, by this agent, on this target, admissible now?* Its content is a set of warrant-bearing assertions, mostly observed, some approved; its operation is the four-valued admissibility function; its maintenance is a continuous lifecycle (Chapter 9).

The boundary plane
: is the pair of substrates of two organisations with disjoint approvers. It answers the question: *is this cross-organisational action admissible on both sides?* Its content is mandates, exposures, attested assertions weighted by trust, and reciprocal commitments; its operation is two-sided admissibility (Chapter 10).

New in Charterion v1.0

The three-plane organisation and the coherence conditions between the planes are introduced by this document. In the research articles the planes were developed separately and their relation was stated in prose. Here the design plane is defined as the *source* of authority, the execution plane as its *enforcement* and the boundary plane as its *portable form*, and six conditions **CC1–CC6** (Chapter 11) make the relation checkable.

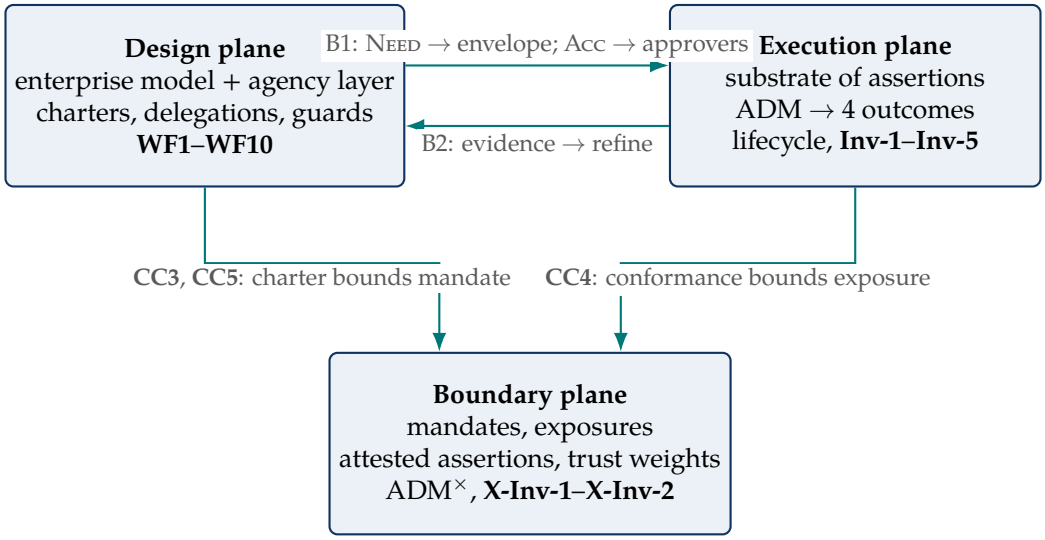

Figure 5.1. The three planes of Charterion and the relations between them. B1 and B2 are the bridges of the method (Chapter 12); **CCn** are coherence conditions (Chapter 11).

5.2 The Agency Grid

The Zachman Framework organises an enterprise description by *who is looking* (rows: perspectives) and *what they ask* (columns: interrogatives). Charterion keeps the second idea and replaces the first: its rows are the three planes—the situations in which a question about agency is asked—and its columns are the five questions of an actor (Section 2.3). Each cell names the constructs that answer the question on that plane. The grid (Table 5.1) is the one-page summary of the framework's content and is used as a viewpoint in its own right (Section 21.4).

Table 5.1. The Agency Grid: planes (rows) against the five questions of an actor (columns)

Plane	Q1 Exists? (anchor)	Q2 Can? (affordance)	Q3 May? (envelope)	Q4 With whom? (coordination)	Q5 How far trusted? (warrant)
Design	Capabilities, processes, steps, resources, humans, roles; Realizes, Step, Requires	Requires(s, x, ℓ): what a step needs of a resource at which level; agentic and reserved steps	Charters, delegations, derived Auth, Covers, Need; **WF1**–**WF3**, **WF6**–**WF8**, **WF10**, **WF12**	Guards: SoD, precedence, oversight, escalation; **WF4**, **WF5**, **WF9**, **WF11**	Approved by construction; validity and review dates; accountability Acc; Accepted-Defect Register
Execution	Anchor assertions: identity, type, part-of, realises, retirement	Affordance assertions: operation, interface, precondition, effect, reversibility	Envelope assertions (approved only): scope, rule, threshold, SoD, escalation condition	Coordination assertions: shared referent, intent, commitment, lock, ownership	Warrant: provenance σ, confidence κ, validity τ, status ε; decision log; *unwarranted*
Boundary	Attested registry identity of counterparty; attested catalogue	External affordance inventory of the exposure	Mandate (outbound); boundary envelope (inbound); ADM$^{\times}$	Reciprocal commitments with shared referent in both logs	Attested status; trust weight $t_A(B)$; κ_{eff}; **X-Inv-2**

5.3 Components

Table 5.2 lists the components of Charterion and where each is specified. The list is the framework's table of contents seen as a product structure, and is the checklist a Conformance Statement refers to.

5.4 Requirements

The framework is designed against twenty-four requirements inherited from the articles and kept unchanged so that traceability to their empirical and theoretical grounding is preserved. They are renumbered by plane: the design-plane requirements **DR1**–**DR6** (the six requirements of the fifth article), the execution-plane requirements **SR1**–**SR12** (the twelve requirements of the Substrate Framework) and the boundary requirements **XR1**–**XR6**. Table 5.3 summarises them; Appendix D traces each to the component that meets it and to the evidence.

New in Charterion v1.0

Version 1.0 adds no requirement to the twenty-four; it adds *conditions* that make several of them checkable across planes (**CC1**–**CC6**) and five well-formedness conditions (**WF8**–**WF12**) that sharpen **DR3** and the oversight requirement. New requirements would be added only by a new version under the policy of Appendix H.

Table 5.2. Components of Charterion v1.0

Component	Content	Where
Content metamodel	Core enterprise model; agency layer; substrate; boundary; coherence conditions; extension profiles	Part II
Formal semantics	Stratified Datalog programme; admissibility function; two-sided admissibility; propositions with proofs	Chapters 7, 9, 10; Appendix A
Method	Design loop D1–D5, operation loop O1–O6, boundary loop X1–X4, bridges B1–B2; agent onboarding; slice discipline	Part III
Governance model	Roles, bodies, decision rights, charter lifecycle, invariants as gates, evidence-gated promotion, regulatory alignment, metrics	Part IV
Artefact framework	Catalogues, matrices, viewpoints, reports; exchange format	Part V
Reference implementation	Reasoner, extension analysis, validator, schema, case models	Part VI
Conformance and maturity	Levels 0–4, exposure levels X0–X4, design-plane conformance, maturity model, Conformance Statement	Chapter 25
Interoperability	ArchiMate specialisation profile; TOGAF ADM mapping; level mapping to external scales	Appendices E and F; Section 8.9

Table 5.3. Requirements of Charterion by plane (summary; full traceability in Appendix D)

Id	Requirement
Design plane (El Hassani, 2026e)	
DR1	Agents as principals: a non-human principal distinct from roles and application components, with an identity to which authority and accountability attach.
DR2	Business-scoped, derived authority: authority stated at capability, process or step and derived to resources through the model, so every entitlement has an architectural justification.
DR3	Traceable accountability: every authority to change state traces to an identified human, directly or along delegations.
DR4	Attenuating delegation: delegated authority never exceeds the delegator's; the delegator's principal remains accountable.
DR5	Design-time coordination: detection before deployment of uncoordinated concurrent writers and of single agents holding both sides of a separation-of-duty pair.
DR6	Tractability: decidable, polynomial-time analysis on models of realistic size, usable as a continuous repository check.
Execution plane (El Hassani, 2026c)	
SR1–SR5	Content: identity; affordance; authority as evaluable rules; coordination through shared referents; warrant on every assertion.
SR6–SR8	Decision: four-valued admissibility; instance-level currency in the acted-upon slice; machine-readable access.
SR9–SR10	Maintenance: agent maintenance under separation of observed and approved; federation with centralised commitment.
SR11–SR12	Governance: non-self-authorisation; accountability of every decision.
Boundary plane (El Hassani, 2026d)	
XR1–XR6	Portable authority; declared exposure; asymmetric warrant; two-sided admissibility with local escalation; reciprocal commitments; joint accountability.

Part II

Content metamodel and semantics

6 The core enterprise model

6.1 Sorts

A Charterion model is built over nine disjoint sorts of constants: humans H, roles R, agents A, capabilities C, processes P, steps S, resources X, charter identifiers M and action levels $\mathbb{L}$. Two derived sorts are used throughout: the sort of *scopes* $\Sigma = C \cup P \cup S$ and the sort of *principals* $\Pi = H \cup R$. Constants are identifiers; names, descriptions and other attributes are carried by the exchange format (Section 21.6) and play no part in the semantics.

6.2 Action levels

Definition 6.1 (Action levels)**.** All authority in Charterion is qualified by a level from the totally ordered set

$$\mathbb{L} = \{\mathit{read} < \mathit{propose} < \mathit{commit} < \mathit{autonomous}\},$$

treated as a finite lattice with meet $\sqcap$ (minimum) and join $\sqcup$ (maximum). At *read* an agent may observe a resource; at *propose* it may prepare a change that a person or another authorised agent commits; at *commit* it may change the resource, each change being confirmed or reversible under a declared procedure; at *autonomous* it may change the resource without per-action confirmation. Levels $\ell \geq \mathit{commit}$ are the *write levels*.

The four levels compress the ten-level scale of Parasuraman et al. (2000) to the distinctions that matter for entitlement provisioning: whether an agent can change state at all, whether a person confirms each change, and whether the change needs confirmation at all. They are used with the same meaning on all three planes, so that a design-time charter, a run-time envelope rule and a portable mandate speak of the same thing (El Hassani, 2026f). Section 8.9 maps them to the finer scales published since 2025.

Remark. Finer distinctions (for instance *commit* with prior confirmation versus *commit* with a reversal window) can be encoded as a longer chain in $\mathbb{L}$ without changing any rule of the semantics, because the rules use only the order, the meet and the join. A *partial* order of levels would require replacing the meet-based attenuation of Chapter 7 by a more general operator and is out of scope for v1.0.

6.3 Extensional predicates

A Charterion model is a finite set of ground facts over the predicates of Table 6.1 (the extensional database, EDB). The core is a deliberate simplification of ArchiMate: it keeps only the elements and relations the agency analysis needs—realisation of capabilities by processes, composition of processes from steps, access of steps to resources, assignment of people to roles—and represents the technology layer solely by the entitlements actually provisioned. Any ArchiMate model can be projected onto this core (Section 6.4).

Table 6.1. Extensional predicates of a Charterion model. Core predicates are inherited from El Hassani (2026e); extension predicates are new in v1.0 and optional.

Layer	Predicate	Meaning
Business	$\textsc{Human}(h)$, $\textsc{Role}(r)$, $\textsc{Holds}(h, r)$	people, roles, and who holds which role
	$\textsc{Capability}(c)$, $\textsc{Process}(p)$, $\textsc{Realizes}(p, c)$	capabilities and the processes that realise them
	$\textsc{Step}(s, p)$, $\textsc{Before}(s, s')$	steps of a process and their control-flow order
	$\textsc{Agentic}(s)$	step s is designated by the business to be performed by an agent
	$\textsc{Reserved}(s)$ *(ext.)*	step s is reserved by the business to human performance
Application	$\textsc{Resource}(x)$, $\textsc{Requires}(s, x, \ell)$	resources (application services and data objects), and the level at which step s needs x
Technology	$\textsc{Grant}(a, x, \ell)$	entitlement of agent a on x at level ℓ as provisioned in identity management
Agency	$\textsc{Agent}(a)$	agent (non-human principal)
	$\textsc{Charter}(m, a, \sigma, \ell, \pi)$	charter m: agent a may act within scope $\sigma \in \Sigma$ up to level ℓ under principal $\pi \in \Pi$
	$\textsc{Delegation}(a, b, \sigma, \ell)$	agent a delegates to agent b within σ up to level ℓ
	$\textsc{SoD}(s, s')$	separation of duty: no single agent may cover both steps
	$\textsc{Precedence}(a, b, x)$	on contention over x, a's actions take precedence over b's
	$\textsc{Oversight}(a, h)$	human h monitors and can halt agent a
	$\textsc{Escalation}(a, h)$ *(ext.)*	human h decides the escalations raised by agent a at run time
	$\textsc{Validity}(m, t_{\text{from}}, t_{\text{until}})$, $\textsc{Review}(m, \rho)$ *(ext.)*	validity interval and review date of charter m (temporal profile)
	$\textsc{Cap}(m, \lambda_{\text{permit}}, \lambda_{\text{escalate}}, u)$ *(ext.)*	quantitative bands of charter m in unit u (quantitative profile)
Application	$\textsc{Irreversible}(x)$ *(ext.)*	writes on resource x have no technical undo (reversibility profile; E1)
	$\textsc{Compensation}(x, s_c)$ *(ext.)*	step s_c compensates writes on x (reversibility profile; E1)

6.4 Projection from ArchiMate

An ArchiMate repository is projected onto the core by a fixed mapping: capabilities to C; business processes to P and their decomposition (composition and aggregation of processes, functions and interactions) to S with triggering and flow relations to Before; business roles and actors to R and H with assignment to Holds; application services and data objects to X; the access and serving relations of a step to Requires, with the access type mapped to a level (read $\mapsto$ *read*; write and read/write $\mapsto$ *commit* by default, raised to *autonomous* only by explicit decision); and realisation of capabilities by processes to Realizes. The agency layer itself is introduced into ArchiMate through the specialisation profile of Appendix E, so that the projection becomes the identity on a repository that already uses the profile. Steps the business intends agents to perform are marked Agentic; steps the business reserves to people are marked Reserved; a step may be neither, in which case its allocation is undecided and the analysis treats it as human-performed but not reserved.

Rationale. The distinction between *not agentic* and *reserved* is new in v1.0 and matters: the granularity study of the fifth article measured "write authority on human steps" as latent authority, a quantity to be minimised; the reserved predicate turns a subset of it into a condition (**WF8**) that must hold. Only the business can say which human steps are reserved by decision (a clinical prescription, an eligibility ruling, a payment release above a threshold) and which are merely not yet delegated.

7 The agency layer

7.1 Constructs

The agency layer adds four element types—agents, charters, delegations and guards—and connects them to the business layer through charter scopes and principals and to the technology layer through the comparison of derived needs with provisioned grants (Figure 7.1).

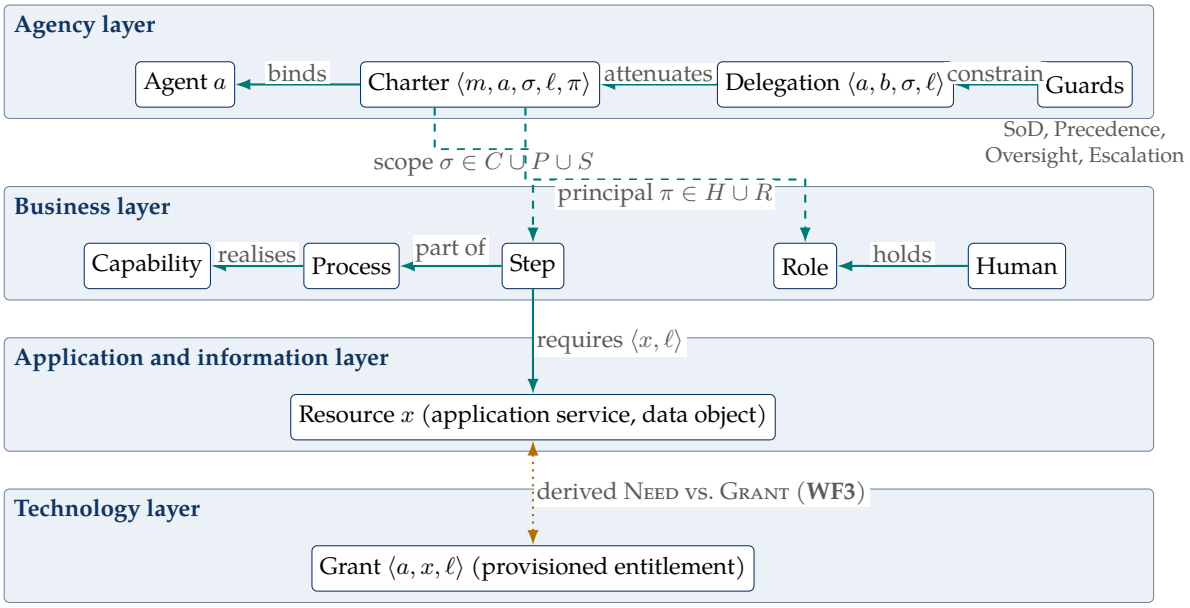

Figure 7.1. Element types and layers of the design plane. Solid arrows are core relations, dashed arrows connect the agency layer to the business layer, the dotted arrow is the comparison performed by the analysis. After El Hassani (2026e), extended with the Escalation guard.

Definition 7.1 (Charter)**.** A *charter* Charter$(m, a, \sigma, \ell, \pi)$ states that agent a may act within business scope σ—a capability, a process or a single step—up to level ℓ, and that principal π (a human, or a role whose holders are humans) is accountable for what a does under it. A charter is a business-level statement made by the organisation, not an entitlement; entitlements are what the analysis derives from charters.

Definition 7.2 (Delegation)**.** A *delegation* Delegation(a, b, σ, ℓ) states that agent a passes on to agent b, within scope σ, authority up to level ℓ. The level is a ceiling, not a grant: the delegatee's derived authority is the meet of the ceiling and the delegator's own derived authority (Proposition 7.2).

Definition 7.3 (Guards)**.** The *guards* are the coordination constructs. SoD(s, s') states that no single agent may cover both steps. Precedence(a, b, x) is the declared resolution of a potential conflict between two agents that may write resource x. Oversight(a, h) names the human who monitors and can halt agent a, and is the condition under which the *autonomous* level is admissible. Escalation(a, h) (new in v1.0) names the human who decides the *escalate* outcomes that the execution plane raises for a; when absent it defaults to the oversight human, and failing that to the humans accountable for a.

Allowing all three scope sorts is deliberate: a charter on a capability is the natural way to express a broad mandate (“the intake agent handles claims”), a charter on a step the natural way to express a narrow one (“the settlement agent may execute payments”), and Section 7.5 states the rule the evidence supports.

7.2 Formal semantics

The semantics of a model is the least model of the stratified Datalog programme of Figure 7.2, evaluated over the EDB together with the built-in order on $\mathbb{L}$. Variables are capitalised, ‘_’ is an anonymous variable, negation is stratified, and comparisons on levels are built-in predicates (Ceri et al., 1989; Abiteboul et al., 1995).

Stratum 0: scopes and process order

(S1) $\textsc{InScope}(\sigma, S) \leftarrow \textsc{Step}(S, \sigma)$. (S2) $\textsc{InScope}(\sigma, S) \leftarrow \textsc{Realizes}(P, \sigma), \textsc{Step}(S, P)$. (S3) $\textsc{InScope}(S, S) \leftarrow \textsc{Step}(S, \_)$.
(O1) $\textsc{Ord}(S, S') \leftarrow \textsc{Before}(S, S')$. (O2) $\textsc{Ord}(S, S'') \leftarrow \textsc{Ord}(S, S'), \textsc{Before}(S', S'')$.
(O3) $\textsc{Conc}(S, S') \leftarrow \textsc{Step}(S, P), \textsc{Step}(S', P'), P \neq P'$.
(O4) $\textsc{Conc}(S, S') \leftarrow \textsc{Step}(S, P), \textsc{Step}(S', P), S \neq S', \neg\textsc{Ord}(S, S'), \neg\textsc{Ord}(S', S)$.

Stratum 1: authority and accountability

(A1) $\textsc{Auth}(A, S, \ell) \leftarrow \textsc{Charter}(\_, A, \sigma, \ell, \_), \textsc{InScope}(\sigma, S)$.
(A2) $\textsc{Auth}(B, S, \ell) \leftarrow \textsc{Delegation}(A, B, \sigma, \ell_d), \textsc{InScope}(\sigma, S), \textsc{Auth}(A, S, \ell_a), \ell = \ell_a \sqcap \ell_d$.
(A3) $\textsc{Auth}_{\geq}(A, S, \ell) \leftarrow \textsc{Auth}(A, S, \ell'), \ell' \geq \ell$.
(K1) $\textsc{Acc}(A, H) \leftarrow \textsc{Charter}(\_, A, \_, \_, H), \textsc{Human}(H)$. (K2) $\textsc{Acc}(A, H) \leftarrow \textsc{Charter}(\_, A, \_, \_, R), \textsc{Holds}(H, R)$.
(K3) $\textsc{Acc}(B, H) \leftarrow \textsc{Delegation}(A, B, \_, \_), \textsc{Acc}(A, H)$.

Stratum 2: coverage and needs

(C1) $\textsc{Shortfall}(A, S) \leftarrow \textsc{Auth}(A, S, \_), \textsc{Requires}(S, X, \ell), \neg\textsc{Auth}_{\geq}(A, S, \ell)$. (C2) $\textsc{Covers}(A, S) \leftarrow \textsc{Auth}(A, S, \_), \neg\textsc{Shortfall}(A, S)$.
(N1) $\textsc{Need}(A, X, \ell) \leftarrow \textsc{Covers}(A, S), \textsc{Requires}(S, X, \ell)$. (N2) $\textsc{Need}_{\geq}(A, X, \ell) \leftarrow \textsc{Need}(A, X, \ell'), \ell' \geq \ell$. (N3) $\textsc{Grant}_{\geq}(A, X, \ell) \leftarrow \textsc{Grant}(A, X, \ell'), \ell' \geq \ell$.

Stratum 3: well-formedness conditions (defects)

(D1) $\textsc{Orphan}(A) \leftarrow \textsc{Auth}(A, \_, \ell), \ell \geq \mathit{commit}, \neg\textsc{HasAcc}(A)$. $\textsc{HasAcc}(A) \leftarrow \textsc{Acc}(A, \_)$.
(D2) $\textsc{Uncovered}(S) \leftarrow \textsc{Agentic}(S), \neg\textsc{Cov}(S)$. $\textsc{Cov}(S) \leftarrow \textsc{Covers}(\_, S)$.
(D3) $\textsc{Shadow}(A, X, \ell) \leftarrow \textsc{Grant}(A, X, \ell), \neg\textsc{Need}_{\geq}(A, X, \ell)$. (D3′) $\textsc{Missing}(A, X, \ell) \leftarrow \textsc{Need}(A, X, \ell), \neg\textsc{Grant}_{\geq}(A, X, \ell)$.
(D4) $\textsc{SoDViol}(A, S, S') \leftarrow \textsc{SoD}(S, S'), \textsc{Covers}(A, S), \textsc{Covers}(A, S')$.
(D5) $\textsc{Contention}(A, B, X) \leftarrow \textsc{Covers}(A, S), \textsc{Covers}(B, S'), A \neq B, \textsc{Requires}(S, X, \ell), \textsc{Requires}(S', X, \ell'), \ell \geq \mathit{commit}, \ell' \geq \mathit{commit}, (\textsc{Conc}(S, S') \vee S = S'), \neg\textsc{Prec}(A, B, X)$.
$\textsc{Prec}(A, B, X) \leftarrow \textsc{Precedence}(A, B, X)$. $\textsc{Prec}(A, B, X) \leftarrow \textsc{Precedence}(B, A, X)$.
(D6) $\textsc{OverDeleg}(A, B, \sigma) \leftarrow \textsc{Delegation}(A, B, \sigma, \ell), \neg\textsc{AuthIn}_{\geq}(A, \sigma, \ell)$. $\textsc{AuthIn}_{\geq}(A, \sigma, \ell) \leftarrow \textsc{InScope}(\sigma, S), \textsc{Auth}_{\geq}(A, S, \ell)$.
(D7) $\textsc{Unguarded}(A) \leftarrow \textsc{Auth}(A, \_, \mathit{autonomous}), \neg\textsc{Watched}(A)$. $\textsc{Watched}(A) \leftarrow \textsc{Oversight}(A, \_)$.

Figure 7.2. The Charterion core semantics as a stratified Datalog programme (El Hassani, 2026e). Levels ℓ range over the finite lattice $\mathbb{L}$; $\sqcap$ is the meet (minimum). The extension rules of Chapter 8 form an additional stratum below.

Stratum 0 computes which steps fall in a scope and which pairs of steps may execute concurrently: steps of different processes always may; steps of the same process may unless one precedes the other in the transitive closure of the declared control-flow order. Stratum 1 derives authority—rule (A1) is the charter, (A2) the delegation with attenuation by the meet, (A3) the upward closure over levels used by later negations—and accountability along charters and delegations (K1)–(K3). Stratum 2 derives coverage (an agent covers a step if it has some authority over it and no requirement of the step exceeds that authority) and the resource-level needs that follow. Stratum 3 evaluates the seven core well-formedness conditions.

7.3 Well-formedness conditions

Definition 7.4 (Well-formed model)**.** A model is *well formed at the core* if the seven conditions **WF1–WF7** of Table 7.1 hold, i.e. if the defect predicates of Stratum 3 are empty in the least model. It is *well formed* if in addition the extension conditions **WF8–WF12** of Chapter 8 (and, in v1.1, **WF13–WF19**) hold for the profiles in use.

Table 7.1. Core well-formedness conditions and the defect reported when each fails

Id	Name	Condition (positive form)	Defect reported
WF1	Accountable authority	Every agent with write-level authority has at least one accountable human	Orphan(a): orphan authority
WF2	Covered designation	Every step designated for agent performance is covered by some agent	Uncovered(s): uncovered step
WF3	Justified provisioning	Provisioned grants equal derived needs	Shadow(a, x, ℓ): grant without justified need; Missing(a, x, ℓ): need without grant
WF4	Separation of duty	No agent covers both steps of a separation-of-duty pair	SoDViol(a, s, s')
WF5	Declared coordination	Any two agents that may write the same resource from concurrent steps have a declared precedence	Contention(a, b, x): unresolved contention
WF6	Bounded delegation	No delegation declares a level above the delegator's authority in its scope	OverDeleg(a, b, σ)
WF7	Watched autonomy	Every agent with autonomous authority is under human oversight	Unguarded(a): unguarded autonomy

WF1 operationalises **DR3**, **WF6** operationalises **DR4**, **WF4** and **WF5** operationalise **DR5**, and **WF3** is the bridge between architecture and provisioning demanded by **DR2**: a grant that no covered step requires is shadow authority the architecture cannot justify, and a need with no grant is an agent that will fail in production. **WF2** is the dual of **WF3** at the business level. **WF7** encodes the oversight requirement of human–automation research and of the AI Act at the level of the architecture: autonomy is not forbidden, but it must be watched.

7.4 Properties

Proposition 7.1 (Least model)**.** *For every finite Charterion model the programme of Figure 7.2 has a unique least model, computable by iterated fixpoint stratum by stratum.*

Proof. The programme is stratified: every negated body predicate belongs to a lower stratum than the rule head, the only recursion—(O2), (A2), (K3)—is positive, and built-in comparisons on the finite lattice are safe because ℓ is bound by a body atom in every rule. Stratified Datalog with safe built-ins over a finite domain has a unique perfect model, which is the least model computed stratum by stratum (Abiteboul et al., 1995). □

Proposition 7.2 (Attenuation). *If* Auth(b, s, ℓ) *is derived through a chain of delegations* $a_0 \to a_1 \to \cdots \to a_k = b$ *from a charter of* a_0 *at level* ℓ_0*, then* $\ell \le \ell_0$ *and* $\ell \le \ell_{d,i}$ *for every delegation level on the chain.*

Proof. By induction on k. For $k = 0$, (A1) gives $\ell = \ell_0$. For $k > 0$, (A2) gives $\ell = \ell_{a_{k-1}} \sqcap \ell_{d,k}$; the meet is below both arguments and, by the induction hypothesis, $\ell_{a_{k-1}}$ is below ℓ_0 and all earlier delegation levels. □

Proposition 7.3 (Accountability preservation). *If* Auth(b, s, ℓ) *is derived through a delegation chain from a charter of* a_0*, then* Acc(a_0, h) *implies* Acc(b, h)*. Consequently a model satisfies* ***WF1*** *whenever every charter that confers a write level names a principal that is a human or a role with at least one holder.*

Proof. The first claim follows from (K3) by induction along the chain. For the second, an agent with write-level authority derives it either from its own charter, whose principal yields Acc(b, h) by (K1) or (K2) under the hypothesis, or through a chain from some a_0 whose charter does, whence Acc(b, h) by the first claim; in both cases (D1) does not fire. □

The proposition identifies the only way **WF1** can fail: a charter whose principal is a role nobody holds (a vacant position, a dissolved team) or a principal absent from the model. Both are common when agents outlive the reorganisation that created them; the cases of Part VII contain several instances.

Proposition 7.4 (Complexity). *Let* n *be the number of facts in a model. The least model is computable in time polynomial in* n*; more precisely in* $O\big(|\textsc{Charter}| \cdot |S| + |\mathbb{L}| \cdot |\textsc{Delegation}| \cdot |S| + |\textsc{Requires}| \cdot |A| + |S| \cdot |\textsc{Before}| + \sum_{x \in X} k_x^2 d_x\big)$*, where* k_x *is the number of agents with a write-level need on* x *and* d_x *bounds the number of write-level step pairs examined per agent pair on* x*.*

Proof. Data complexity of stratified Datalog is polynomial (Dantsin et al., 2001) and the programme is fixed. For the refined bound: (A1) touches each charter–step pair in scope once; (A2) raises Auth$(b, s, \cdot)$ at most $|\mathbb{L}| - 1$ times per delegation–step pair because levels only increase along the fixpoint; (C1)–(N1) examine each requirement against each agent with authority on its step; the closure (O2) is bounded by $|S| \cdot |\textsc{Before}|$; and (D5) examines pairs of agents per resource, whence the quadratic term. □

The quadratic term is intrinsic to the contention condition, which is a property of pairs; in practice k_x is small, and the reference reasoner analyses models of half a million facts in under two seconds (Chapter 26).

7.5 The granularity rule

The scope of charters governs how much latent authority a model contains. The granularity study of the fifth article (El Hassani, 2026e) re-scoped the charters of twenty synthetic models of sixty processes to the capability, the process or a single step: capability charters covered every agentic step but conferred write-level authority over roughly six hundred steps reserved to people and produced two orders of magnitude more contention pairs than step charters; step charters produced almost no latent authority or contention but left more than eighty agentic steps uncovered; process charters sat between. The rule the framework draws from this, and which the cases of Part VII confirm, is stated as a recommendation:

Granularity rule

Charters at *read* and *propose* MAY be broad (capability or process). Charters at *commit* and *autonomous* SHOULD be as narrow as the steps that need them, and a write-level charter on a capability SHALL be recorded in the Charter Rationale Record with the reason a narrower scope was not possible. The analysis makes the cost of each choice—latent authority, reserved-step intrusion, contention—visible before it is provisioned.

8 Extensions of the design plane

New in Charterion v1.0

Everything in this chapter is introduced by v1.0. The extensions add predicates and a fourth stratum of rules; they never alter the core programme of Figure 7.2, so every proposition of Section 7.4 continues to hold, and a model that uses no extension predicate has exactly the semantics of the fifth article. Each extension is a *profile* that a Conformance Statement declares in use; **WF8**, **WF9** and **WF11** are required from design-plane conformance level D2 upward and **WF10** and **WF12** from D3 (Chapter 25).

8.1 Reserved steps and WF8

Definition 8.1 (Reserved step)**.** $\textsc{Reserved}(s)$ states that the business has decided that step s is performed by a human and not by an agent. A step SHALL NOT be both $\textsc{Agentic}$ and $\textsc{Reserved}$; a step may be neither.

Condition 8.1 (**WF8**: Reserved-step integrity)**.** No agent holds write-level authority over a reserved step:

$$\text{(D8)}\quad \textsc{Intrusion}(A,S) \leftarrow \textsc{Reserved}(S),\ \textsc{Auth}(A,S,\ell),\ \ell \geq \mathit{commit}.$$

Rationale. A broad charter confers authority over every step in its scope, including steps the business never meant an agent to touch. The core semantics counts this as *latent authority*, a quantity the granularity study showed to be large under capability charters; it does not treat it as a defect, because the business may simply not have decided. **WF8** lets the business decide, step by step, and turns the decision into a checkable condition. It is also the condition that a regulator asks for first: that consequential decisions—a prescription, an eligibility ruling, a payment release above a threshold, the filing of a suspicious-activity report—are made by people, and that this can be shown from the architecture rather than from a policy document (European Parliament and Council, 2024; IMDA, 2026). Note that read and propose authority over a reserved step are permitted: an agent may prepare the file on which a human decides.

Proposition 8.1 (Intrusion is caused by scope, not by delegation)**.** *If* $\textsc{Intrusion}(b,s)$ *holds and* b*'s authority on* s *is derived through a delegation chain from a charter of* a_0*, then* $\textsc{Intrusion}(a_0,s)$ *holds. Consequently* ***WF8*** *holds for all agents whenever it holds for every agent's own charters.*

Proof. By Proposition 7.2 the level derived for b is at most the level of a_0's charter on s, so a_0 holds authority $\geq$ *commit* on s too. □

The proposition means that **WF8** defects are repaired where they arise: by narrowing or splitting the charter whose scope contains the reserved step, never by editing delegations.

8.2 Escalation reachability, oversight span and WF9

Definition 8.2 (Escalation human, reachable agent, oversight span)**.** The *escalation human* of an agent is given by $\textsc{Escalation}(a,h)$; when no such fact exists it is every h with $\textsc{Oversight}(a,h)$,

and when neither exists it is every h with $\textsc{Acc}(a, h)$:

$$\begin{aligned}
&\text{(E1)}\ \ \textsc{Esc}(A, H) \leftarrow \textsc{Escalation}(A, H),\ \textsc{Human}(H).\\
&\text{(E2)}\ \ \textsc{Esc}(A, H) \leftarrow \neg\textsc{HasEsc}(A),\ \textsc{Oversight}(A, H),\ \textsc{Human}(H).\\
&\qquad\ \ \textsc{HasEsc}(A) \leftarrow \textsc{Escalation}(A, _).\\
&\text{(E3)}\ \ \textsc{Esc}(A, H) \leftarrow \neg\textsc{HasEsc}(A),\ \neg\textsc{Watched}(A),\ \textsc{Acc}(A, H).
\end{aligned}$$

An agent is *reachable* if it has at least one escalation human: $\textsc{Reachable}(A) \leftarrow \textsc{Esc}(A, \_)$. The *oversight span* of a human is the number of agents for which the human is an escalation or oversight human: $\mathrm{Span}(h) = |\{a : \textsc{Esc}(a, h) \vee \textsc{Oversight}(a, h)\}|$. The organisation declares a *span bound* $\beta \in \mathbb{N}$ (default 12).

Condition 8.2 (**WF9**: Escalation reachability)**.** Every agent with write-level authority is reachable, and no human's span exceeds the bound:

$$\begin{aligned}
&\text{(D9)}\ \ \textsc{Unreachable}(A) \leftarrow \textsc{Auth}(A, _, \ell),\ \ell \geq \textit{commit},\ \neg\textsc{Reachable}(A).\\
&\text{(D9}')\ \ \textsc{Overloaded}(H) \leftarrow \mathrm{Span}(H) > \beta.
\end{aligned}$$

Rationale. **WF1** guarantees that some human is *accountable*; **WF9** guarantees that some human is *available*: a person who will actually receive the escalations the execution plane raises (Chapter 9) and who is not so overloaded that receiving them is a formality. The first part closes the vacancy gap that **WF1** leaves for agents whose principal is a role with holders but whose oversight names a role nobody holds. The second part is the architectural expression of the automation-bias finding of human–automation research (Parasuraman et al., 2000) and of the Singapore framework's insistence that human checkpoints be meaningful (IMDA, 2026): a person watching a hundred agents watches none. The default bound of twelve is a placeholder; the organisation SHALL set β in its Conformance Statement and SHOULD justify it from the expected escalation volume per agent.

8.3 The temporal profile and WF10

Definition 8.3 (Validity, review, time-indexed model)**.** $\textsc{Validity}(m, t_{\text{from}}, t_{\text{until}})$ gives the half-open interval $[t_{\text{from}}, t_{\text{until}})$ during which charter m is in force; $\textsc{Review}(m, \rho)$ gives the date by which m must be re-approved. The same predicates apply to delegations, identified by their tuple. At an instant t, charter m is *current* if $t \in [t_{\text{from}}, t_{\text{until}})$, *expired* if $t \geq t_{\text{until}}$, and *stale* if it is current and $\rho < t$. The *time-indexed model* M_t is obtained from M by removing every expired charter and delegation and by replacing the level ℓ of every stale charter and delegation by its predecessor $\mathrm{pred}(\ell)$ in $\mathbb{L}$ (*autonomous*→*commit*→*propose*→*read*; $\mathrm{pred}(\textit{read}) = \textit{read}$). The semantics of M at t is the least model of M_t.

Condition 8.3 (**WF10**: Temporal currency)**.** Every write-level charter has a finite validity and a review date, and at the instant of evaluation no write-level charter is expired or stale:

$$\begin{aligned}
&\text{(D10)}\ \ \textsc{Unbounded}(M) \leftarrow \textsc{Charter}(M, _, _, \ell, _),\ \ell \geq \textit{commit},\\
&\qquad\qquad (\neg\textsc{HasValidity}(M)\ \vee\ \neg\textsc{HasReview}(M)).\\
&\text{(D10}')\ \ \textsc{Expired}(M) \leftarrow \textsc{Charter}(M, _, _, \ell, _),\ \ell \geq \textit{commit},\ \textsc{Validity}(M, _, t_u),\ t \geq t_u.\\
&\text{(D10}'')\ \ \textsc{Stale}(M) \leftarrow \textsc{Charter}(M, _, _, \ell, _),\ \ell \geq \textit{commit},\ \neg\textsc{Expired}(M),\\
&\qquad\qquad \textsc{Review}(M, \rho),\ \rho < t.
\end{aligned}$$

The same three rules apply to write-level delegations. UNBOUNDED is a modelling defect (the profile is in use but a charter carries no dates); EXPIRED and STALE are evaluated at the instant t of analysis, and their witnesses are exactly the charters that M_t removes or degrades.

Proposition 8.2 (Temporal evaluation preserves the core properties)**.** *For every instant t, Propositions 7.1, 7.2, 7.3 and 7.4 hold of the least model of M_t. Moreover, for all a, s: $\textsc{Auth}_{M_t}(a, s, \ell)$ implies $\textsc{Auth}_M(a, s, \ell')$ for some $\ell' \geq \ell$; that is, degradation and expiry never create authority.*

Proof. M_t is a Charterion model, so the propositions apply to it directly. For the second claim, M_t is obtained from M by deleting facts and lowering charter and delegation levels; (A1) and (A2) are monotone in the levels of their body atoms and in the presence of facts, so every authority derived in M_t is derived in M at a level at least as high. □

Corollary 8.1 (Degradation is safe and visible)**.** *Passing from M to M_t cannot introduce defects of kinds **WF4**, **WF5**, **WF7** or **WF8**, all of which require authority; it can introduce defects of kinds **WF2** and **WF3**-missing, which signal that a stale or expired charter no longer sustains a step or an entitlement the organisation relies on.*

Proof. The first four defects are monotone in Auth and Covers, which only shrink; Uncovered and Missing are anti-monotone in them. □

Rationale. The temporal profile gives the design plane the property that **Inv-4** gives every assertion of the execution plane: a finite validity, and a refusal to answer confidently beyond it. Its distinctive feature is *graceful degradation*: a charter that has passed its review date does not vanish—which would halt the agents that depend on it and punish the business for the architecture team's backlog—but loses one level, so that an unreviewed autonomous agent becomes a commit agent whose every change is confirmed, and an unreviewed commit agent becomes a proposer. The degradation is visible in the model as a Stale defect and in operations as confirmations that were not requested before, both of which create the pressure to review. Corollary 8.1 guarantees that the degraded model is never more dangerous than the reviewed one. The fifth article listed a temporal extension as future work; this profile is its first form and, unlike the core, has not been evaluated experimentally (Chapter 26).

8.4 Independent separation and WF11

New in Charterion v1.0

Separation of duty was conceived for people: two individuals do not share a mind, so requiring two of them for a sensitive pair of steps buys independence. Two agents do not have that property. When an orchestrator delegates one side of a pair to a sub-agent and performs the other itself, or delegates both sides to two sub-agents, **WF4** is satisfied—no single agent covers both steps—and the separation is void, because one planner controls both executions. The underlying problem is known: the satisfiability of separation-of-duty constraints in the presence of delegation was analysed for workflow authorisation systems by Crampton and Khambhammettu (2008) and Wang and Li (2010), after Crampton (2005) had shown how constraints on task pairs propagate through a workflow. What is new here is not the result but its transport to the enterprise model—agents, charters, orchestrators and delegation chains in place of users, roles and workflow steps—and its statement as a design-time condition that none of the 2025–2026 agentic-governance frameworks states.

Definition 8.4 (Dependence)**.** Let $\textsc{Del}^*$ be the transitive closure of the delegation relation on agents, $\textsc{Del}^*(A, B) \leftarrow \textsc{Delegation}(A, B, \_, \_)$; $\textsc{Del}^*(A, C) \leftarrow \textsc{Del}^*(A, B), \textsc{Delegation}(B, C, \_, \_)$. Two distinct agents a, b are *dependent*, $\textsc{Dep}(a, b)$, if $\textsc{Del}^*(a, b)$ or $\textsc{Del}^*(b, a)$ (one controls the other along a chain) or $\exists c : \textsc{Del}^*(c, a) \wedge \textsc{Del}^*(c, b)$ (a common delegator controls both).

Condition 8.4 (**WF11**: Independent separation)**.** The two sides of a separation-of-duty pair are covered only by pairwise independent agents:

(D11) $\textsc{Collusion}(A, B, S, S') \leftarrow \textsc{SoD}(S, S'),\ \textsc{Covers}(A, S),\ \textsc{Covers}(B, S'),\ A \neq B,\ \textsc{Dep}(A, B).$

Rationale. **WF4** and **WF11** together restore to agent populations the property that separation of duty had for people. **WF11** is deliberately restricted to dependence through *delegation*, which is the relation by which one agent's plan becomes another agent's action. Two agents that share a *principal* (the same human is accountable for both) are not dependent in this sense: the human is exactly the independent judgement the pair was meant to involve, and many organisations will legitimately charter both sides of a pair under one department head. The analysis nonetheless reports shared-principal pairs as an informative metric (SharedPrincipal), so that the Agency Governance Board can decide whether a given pair should additionally require distinct principals; where it does, the organisation records the requirement as a SoD on principals in the Guard Register.

Proposition 8.3 (Removal of collusion)**.** *Let M be a model and $D \subseteq \textsc{Delegation}^M$ any set of delegations (for instance those on the dependence paths of the Collusion pairs). There is a model M' obtained from M by removing D and adding, for every tuple $\textsc{Auth}^M(g, s, \ell)$ no longer derivable, one charter $\textsc{Charter}(m', g, s, \ell, \pi)$ with π a principal of g's own charters that resolves to a human, or else a human of $\textsc{Acc}^M(g)$, such that (i) $\textsc{Auth}^{M'} = \textsc{Auth}^M$, hence Covers, Need and every defect derived from them are unchanged; (ii) every agent with an accountable human in M has one in M'; (iii) $\textsc{Dep}^{M'} \subseteq \textsc{Dep}^M$, so $\textsc{Collusion}^{M'} \subseteq \textsc{Collusion}^M$, and a pair whose every dependence path meets D is no longer in $\textsc{Collusion}^{M'}$. In particular every collusion pair can be removed without changing the coverage of any agentic step.* (Restated in v1.1, E15: the v1.0 statement left the recharter step and the case of cyclic delegation graphs implicit.)

Proof. (i) Removing D can only remove Auth tuples (Proposition 7.1: the least model is monotone in the EDB for the positive strata 0–1). For each removed tuple (g, s, ℓ) the added charter derives exactly it by (A1) with $\textsc{InScope}(s, s)$ (S3); a charter on a single step derives no other Auth tuple, and every tuple derivable in M through a delegation not in D is still derivable. Hence $\textsc{Auth}^{M'} = \textsc{Auth}^M$, and strata 2–3 depend on the EDB only through Auth, Requires, Grant, SoD, Precedence and Oversight, which are unchanged. (ii) The principal chosen for each added charter resolves to a human by construction, and g's other accountability derivations are unaffected by removing delegations unless they passed through D, in which case the added charter supplies one by (K1)–(K2); an agent whose Auth tuples were all unaffected keeps its Acc derivations from its own charters, and an agent that lost an Acc derivation through D but no Auth tuple is rechartered once on any step it covers. (iii) $\textsc{Del}^*$ is the closure of a smaller relation, so Dep shrinks or stays; Collusion is Dep joined with unchanged relations. The construction is a single pass over finitely many tuples and needs no repetition, which covers cyclic delegation graphs as well as acyclic ones. The reference implementation is `charterion_v11.collusion_repair`; on the three first-pass cases with collusion (Table 23.8) it removes every pair with Auth, Covers, Need, accountability and all other defects unchanged (`results/e15_check.json`). □

The five cases of Part VII show the pattern: three of the five first-pass models contain a collusion pair, in each case an orchestrator that delegates one side of a regulated pair to a specialist agent while retaining authority over the other; all are removed by iteration 2 without loss of coverage (Table 23.8).

8.5 Compensable autonomy and WF12

New in Charterion v1.0

The action-level lattice distinguishes *commit*, where a change is confirmed or reversible under a declared procedure, from *autonomous*, where it is not confirmed. Nothing so far distinguishes changes that *can* be reversed from changes that cannot—a payment settled, a message sent, a record erased, a contract accepted. Autonomy scales published in 2026 treat reversibility as a dimension of risk; Charterion makes it a condition on the model: autonomous authority over an irreversible operation is admissible only where the enterprise model contains the compensating step, so that the organisation has decided, in the architecture, how the consequence will be undone or absorbed.

Definition 8.5 (Irreversibility and compensation)**.** $\textsc{Irreversible}(x)$ states that write-level operations on resource x cannot be reversed by the actor that performed them. $\textsc{Compensation}(x, s_c)$ states that step s_c is the organisation's compensating action for such operations (a refund step, a recall step, a correction entry, a customer notification with a right of withdrawal). A compensation step is *available* if it is covered by some agent or reserved to a human: $\textsc{Avail}(S_c) \leftarrow \textsc{Covers}(\_, S_c)$; $\textsc{Avail}(S_c) \leftarrow \textsc{Reserved}(S_c)$.

Condition 8.5 (**WF12**: Compensable autonomy)**.** No agent holds autonomous authority over a step that writes an irreversible resource unless a compensation for that resource is available:

$$\begin{aligned}(\text{D12})\ \textsc{Uncompensated}(A, X, S) \leftarrow\ & \textsc{Auth}(A, S, \textit{autonomous}),\ \textsc{Requires}(S, X, \ell),\ \ell \geq \textit{commit},\\ & \textsc{Irreversible}(X),\ \neg\textsc{HasComp}(X);\\ \textsc{HasComp}(X) \leftarrow\ & \textsc{Compensation}(X, S_c),\ \textsc{Avail}(S_c).\end{aligned}$$

Rationale. Compensation is not new: it is the mechanism of Sagas (Garcia-Molina and Salem, 1987) and of the compensation handlers of WS-BPEL (OASIS, 2007), and reversibility is a dimension of the 2026 autonomy scales (Chapter 24). What **WF12** adds is to make compensability a *checkable enterprise-architecture precondition for autonomous authority*: it does not forbid autonomy on irreversible operations—a payments organisation may well want a settlement agent that does not wait for a person—but requires that the architecture, not the incident review, name what happens next, and that the compensating step be actually performable (covered or reserved) at the instant the autonomy is conferred. It also gives the run time something to enforce: the charter-derived envelope (Definition 9.5) carries, for every autonomous entry on an irreversible resource, the identifier of the compensation step, so that a reversal recorded in the decision log (**CC6**) is attributable to a step of the model. Irreversible and Compensation are extension predicates; where the profile is not in use **WF12** is vacuous, and a Conformance Statement at design-plane level D3 or above SHALL declare whether the profile is used for every slice that contains an *autonomous* charter. On the five cases (Section 23.8) the profile finds uncompensated autonomy in three organisations, in two of which no reversal step exists in the model at all—the finding is then that a step is missing, not that a charter is wrong.

Table 8.1 summarises the five conditions added by v1.0 to the seven of the fifth article.

8.6 The quantitative profile

Definition 8.6 (Charter caps)**.** $\textsc{Cap}(m, \lambda_{\text{permit}}, \lambda_{\text{escalate}}, u)$ attaches to a write-level charter m two thresholds in unit u (a currency, a count, a duration): below λ_{permit} the agent may act at the charter's level; between λ_{permit} and $\lambda_{\text{escalate}}$ the action is subject to a decision by the escalation human; above $\lambda_{\text{escalate}}$ it is denied. A charter without a cap has no quantitative limit.

Table 8.1. Well-formedness conditions added by Charterion v1.0

Id	Name	Failure it detects	Defect	Profile
WF8	Reserved-step integrity	write-level authority over a step the business reserved to humans	Intrusion	reserved
WF9	Escalation reachability	agent whose escalate outcomes reach no current human; humans overseeing more agents than the bound	Unreachable, Overloaded	escalation
WF10	Temporal currency	write-level charters without dates, expired, or past review	Unbounded, Expired, Stale	temporal
WF11	Independent separation	separation-of-duty pair whose two agents are controlled through one delegation chain	Collusion	core (no extra facts)
WF12	Compensable autonomy	autonomous authority over a step writing an irreversible resource without an available compensating step	Uncompensated	reversibility

The caps do not enter the design-plane semantics, which has no quantities; they are carried into the execution plane as threshold rules of the envelope (Section 9.7) and bound the bands of any portable mandate issued to the agent (**CC3**, Chapter 11). Their place in the design plane is to make the organisation's quantitative decision an architectural fact recorded once, with a principal, rather than a parameter set differently in every policy engine.

8.7 Reach and blast radius

Two derived quantities are used by the governance model (Chapter 20) and reported in the Autonomy & Oversight viewpoint.

Definition 8.7 (Reach, blast radius)**.** The *reach* of an agent is the set of resources it may write: $\mathrm{Reach}(a) = \{x : \textsc{Need}_{\geq}(a, x, \mathit{commit})\}$. Its *blast radius* is the reach of everything it can delegate to, transitively: $\mathrm{Blast}(a) = \mathrm{Reach}(a) \cup \bigcup_{b:\textsc{Delegation}(a,b,\_,\_)} \mathrm{Blast}(b)$.

By Proposition 7.2, a delegatee's reach within a delegated scope is bounded by the delegator's; the blast radius nevertheless exceeds the reach whenever a delegatee holds charters of its own, which is exactly the situation in which a compromised orchestrator can reach resources its own charters never named. The blast radius is the design-time quantity that the runtime-security literature calls the *agency perimeter* (Amazon Web Services, 2025); Charterion computes it from the model rather than inferring it from traffic.

8.8 Ontological grounding (informative)

The constructs of the agency layer can be grounded in the Unified Foundational Ontology (UFO) in the manner that Azevedo et al. (2015) grounded ArchiMate's resources and capabilities. An *agent* is an agentive object that is not a person; a *charter* is a social relator that mediates the agent and its principal and founds two modes: the agent's *permission* within the scope and the principal's *accountability*; a *delegation* is a relator between two agents founded on the delegator's permission and therefore existentially dependent on it (which is what attenuation expresses); *guards* are normative descriptions in the sense of UFO-C; and the *action levels* qualify the permission mode. The grounding explains why a charter is not an entitlement (a relator is not a property of one relatum) and why accountability propagates along delegation (the dependent relator inherits the founding relator's accountable party). A full axiomatisation is future work; Chapter 26 lists it among the open items.

8.9 Mapping of action levels to external scales

Several autonomy scales have been published since 2025. Table 8.2 maps Charterion's four levels onto them so that an organisation using one of those scales for policy can express its charters without translation loss. The mapping is the author's reading of the public documents and is informative; the CSA framework grades autonomy on five dimensions (decision authority, scope, reversibility, impact, temporal duration) of which Charterion's single level axis corresponds most closely to decision authority and reversibility, so the correspondence is indicative only.

Table 8.2. Correspondence between Charterion action levels and external autonomy scales (informative)

Charterion level	Parasuraman et al. (2000) (10 levels)	CSA Autonomy Levels & Control Framework v2.0, six levels L0–L5 (Cloud Security Alliance, 2026b) (indicative)	Meaning in Charterion
read	1–2: computer offers no assistance or a complete set of alternatives	L1 (indicative)	observe only; broad charters acceptable
propose	3–4: narrows or suggests one alternative	L2 (indicative)	prepare a change another principal commits
commit	5–6: executes if human approves, or allows veto within a time window	L3 (indicative)	change state with confirmation or reversal procedure
autonomous	7–9: executes and informs, on request, or if it decides to	L4 (indicative)	change state without per-action confirmation; requires Oversight (**WF7**)
—	10: decides everything, ignores the human	L5 (indicative)	not admissible in Charterion: **WF7** requires oversight and **WF9** requires reachability

8.10 Stratum 5 and the preservation of the v1.0 semantics

New in Charterion v1.1

Version 1.1 adds seven optional extension profiles—configuration, instance, staffing, cap-composition, obligation, perimeter and containment—with the conditions **WF13–WF19**, and one coherence condition, **CC7** (Section 11.5). Every rule they introduce belongs to a single new stratum, *stratum 5*, placed below stratum 4 of Figure 7.2 and Chapter 8. Under Appendix H the semantics of every v1.0 predicate, condition and algorithm is unchanged; this section states and proves the property once, so that each profile only has to exhibit its rules.

Let $\Pi = \Pi_0 \cup \cdots \cup \Pi_4$ be the programme of Figure 7.2 together with the extension rules of Sections 8.1–8.5, and let Π_5 be the set of rules introduced in Sections 8.11–8.17 and Section 11.5.

Proposition 8.4 (Stratum 5 preserves the v1.0 semantics). (*i*) *$\Pi \cup \Pi_5$ is a stratified programme with the stratum order $0 < 1 < 2 < 3 < 4 < 5$: every head predicate of Π_5 is new, occurs in no rule of Π, and every body of Π_5 reads extensional predicates, derived predicates of strata ≤ 4, aggregates (counts, sums, shortest paths) over finite relations of strata ≤ 4 or of the EDB, and negation only over extensional predicates and non-recursive stratum-5 auxiliaries.*

(*ii*) *For every model M and every set F of profile facts, the least model of $\Pi \cup \Pi_5$ on $M \cup F$ restricted to the predicates of Π equals the least model of Π on M. In particular Propositions 7.1–7.4, 8.1, 8.2, 8.3 and Corollary 8.1 hold verbatim.*

(*iii*) *No rule of Π_5 has Charter, Delegation, Grant, Auth, Auth$_{\geq}$, Covers, Need or Need$_{\geq}$ in its head; hence $\textsc{Auth}^{M \cup F} = \textsc{Auth}^{M}$ and $\textsc{Need}^{M \cup F} = \textsc{Need}^{M}$.*

(*iv*) *The complete v1.1 analysis is polynomial in the size of the model, the profile facts and the registers: to the bound of Proposition 7.4 and Proposition 11.1 it adds the terms stated in Propositions 8.5–8.11 and 11.4, each linear or join-linear in the profile facts.*

Proof. (i) is verified rule by rule in Sections 8.11–8.17 and Section 11.5; the profile predicates (Config, Reviewed, Instance, Cardinality, Performs, DelegationCap, Obligation, Collaboration, BlastBound, AgentClass, ClassBlastBound) are extensional and the defect predicates are new. (ii) Stratified evaluation computes strata bottom-up and a stratum reads only lower strata; adding Π_5 above stratum 4 therefore leaves the fixpoint of strata 0–4 untouched, and the profile facts occur only in bodies of Π_5. (iii) By inspection of the heads. (iv) Each stratum-5 rule is a join of profile facts against indexed derived relations, a count, a sum or a breadth-first search, bounded as stated in the cited propositions. □

Corollary 8.2 (No profile creates authority)**.** *For every v1.1 profile and every model, the charter-derived envelope (Definition 9.5) and the mandates grounded in charters (Section 10.4) are the same with and without the profile's facts. A profile can report a defect; it cannot confer, widen or extend an authority.* ***Inv-3***, ***X-Inv-1*** *and Proposition 16.1 are therefore unaffected by the additions of v1.1.*

Proof. Proposition 8.4(iii): the envelope is a function of Need and mandates are bounded by Need and Cap (**CC3**), none of which a stratum-5 rule derives. □

Remark. Every condition of v1.1 is *vacuous when its profile is not in use*: each rule has a positive literal of a profile predicate in its body, or is guarded by the declaration of the profile in the Conformance Statement (Section 25.4, item 3), following the convention of **WF12**. A model conformant to v1.0 at a level is therefore conformant to v1.1 at the same level (Appendix H); the regression report of the toolkit (Chapter 22) verifies this mechanically on every v1.0 test and case model.

8.11 The configuration profile and WF13

New in Charterion v1.1

The agent that was reviewed at gate G2 is not necessarily the agent that runs: model, prompt, tool set and runtime version change without a change to the charter. The configuration profile records the configuration digest reviewed with each charter and the digest actually running, and reports their divergence. It binds to the OWASP Agent Control Standard's agent bill of materials (AG-BOM; Appendix K).

Definition 8.8 (Configuration digest)**.** A *configuration digest* is an opaque identifier κ of the running configuration of an agent—model identity and version, system prompt, tool set, runtime version—such that two configurations with different behaviour-relevant content have different digests. The framework does not fix the digest function; a binding shall state it.

Configuration profile: predicates and rules (stratum 5)

Extensional: $\textsc{Config}(a, \kappa)$ — agent a runs configuration κ (observed at O1); $\textsc{Reviewed}(m, \kappa)$ — κ was reviewed when charter m was approved (G2).

(D13a)
$\textsc{ConfigDrift}(A, M) \leftarrow \textsc{Charter}(M, A, \_, \ell, \_),\ \ell \geq \mathit{commit},\ \textsc{Reviewed}(M, \kappa_r),\ \textsc{Config}(A, \kappa),\ \kappa \neq \kappa_r.$

(D13b) $\textsc{Unreviewed}(M) \leftarrow \textsc{Charter}(M, \_, \_, \ell, \_),\ \ell \geq \mathit{commit},\ \neg\exists\kappa\, \textsc{Reviewed}(M, \kappa).$

Evaluated on the time-indexed model M_t (charters in force at t).

Condition 8.6 (**WF13**: Configuration currency)**.** When the configuration profile is in use, every write-level charter in force SHALL carry a reviewed configuration digest, and the running configuration of its agent SHALL equal it: $\textsc{Unreviewed} = \emptyset$ and $\textsc{ConfigDrift} = \emptyset$. Defects: $\textsc{Unreviewed}(m)$, $\textsc{ConfigDrift}(a, m)$.

Rationale. **WF1–WF12** say nothing about whether the agent that runs is the agent that was reviewed. **WF13** makes the identity of the reviewed and the running configuration a checkable fact. It does not degrade authority on drift: a degradation would compose with the temporal profile and is a v2.0 candidate (Appendix J). Governance consequence (SHOULD): while $\textsc{ConfigDrift}(a, m)$ holds the charter is *Under review* (Table 18.4), its envelope entries are flagged in the Envelope Register and no new mandate is issued from it.

Proposition 8.5 (**WF13**)**.** *Given M_t and its least model, D13a–D13b are evaluable in $O(|\textsc{Charter}| + |\textsc{Reviewed}| + |\textsc{Config}|)$ with hash-indexed lookups, and create no authority.*

Proof. Each charter is examined once against two constant-time lookups; Corollary 8.2. □

8.12 The instance profile and WF14

New in Charterion v1.1

An agent type is chartered once and instantiated many times, ephemerally. The core semantics is stated on types; the instance profile records running instances, bounds their number and counts the oversight span over instances. An instance-aware core is a v2.0 candidate (Appendix J).

Definition 8.9 (Instance)**.** An *instance* i of agent a is a running process that acts under the identity of a and holds no charter of its own. Instances inherit the charters, delegations and guards of their type by construction: every rule of Figure 7.2 is stated on types, and the anchor layer of the substrate records $\textsc{Instance}(i, a)$.

Instance profile: predicates and rules (stratum 5)

Extensional: $\textsc{Instance}(i, a)$; $\textsc{Cardinality}(a, n)$ — at most n concurrent instances of a.
$\textsc{Chartered}(A) \leftarrow \textsc{Charter}(\_, A, \_, \_, \_)$. $\textsc{Chartered}(B) \leftarrow \textsc{Delegation}(\_, B, \_, \_)$.
(D14a) $\textsc{Uncharted}(I) \leftarrow \textsc{Instance}(I, A),\ \neg\textsc{Chartered}(A)$.
(D14b) $\textsc{OverInstantiated}(A) \leftarrow \textsc{Cardinality}(A, n),\ \#\{I : \textsc{Instance}(I, A)\} > n$.
$\textsc{Inst}(A) = \max(1, \#\{I : \textsc{Instance}(I, A)\})$; $\textsc{SpanI}(H) = \sum_{A:\, \textsc{Watches}(H,A)} \textsc{Inst}(A)$, where $\textsc{Watches}(H, A)$ holds when H is a human of an $\textsc{Oversight}(A, \cdot)$ or $\textsc{Escalation}(A, \cdot)$ fact (as in Section 8.2).
(D14c) $\textsc{OverloadedI}(H) \leftarrow \textsc{SpanI}(H) > \beta$.

Condition 8.7 (**WF14**: Chartered instantiation)**.** When the instance profile is in use, every running instance SHALL be an instance of a chartered or delegated-to type, no type SHALL exceed its declared cardinality, and the oversight span counted over instances SHALL not exceed β. Defects: $\textsc{Uncharted}(i)$, $\textsc{OverInstantiated}(a)$, $\textsc{OverloadedI}(h)$.

Rationale. **WF9** bounds the span over types; ten instances of one autonomous type under one human are ten things to halt. UNCHARTED is the run-time dual of **WF1**: an agent retired in the model but still running (UC5, iteration 2, Table 23.12).

Proposition 8.6 (**WF14**)**.** *D14a–D14c are evaluable in $O(|\textsc{Instance}| + |\textsc{Cardinality}| + |\textsc{Oversight} \cup \textsc{Escalation}| \cdot \bar{h})$, where $\bar{h}$ bounds the humans a role resolves to; they create no authority; SPANI = SPAN when every type has exactly one instance.*

Proof. Counting and summation over indexed relations; Corollary 8.2; $\textsc{Inst}(A) = 1$ for the last claim. □

8.13 The staffing profile and WF15

New in Charterion v1.1

WF2 says every agentic step is covered by an agent; **WF8** keeps agents off reserved steps. Nothing in v1.0 checks that a reserved step can be performed at all. **WF15** is the dual of **WF2** on the human side: every reserved step has a role assigned to it that has a current holder.

Staffing profile: predicates and rules (stratum 5)

Extensional: $\textsc{Performs}(r, s)$ — role r is assigned to perform step s.
$\textsc{Staffed}(S) \leftarrow \textsc{Performs}(R, S),\ \textsc{Holds}(H, R),\ \textsc{Human}(H).$
(D15) $\textsc{UnstaffedReserved}(S) \leftarrow \textsc{Reserved}(S),\ \neg\textsc{Staffed}(S).$
Metric (informative): $\textsc{ReservedLoad}(H) = \#\{S : \textsc{Reserved}(S),\ \textsc{Performs}(R, S),\ \textsc{Holds}(H, R)\}.$

Condition 8.8 (**WF15**: Staffed reservation)**.** When the staffing profile is in use, every reserved step SHALL have a role assigned to perform it with at least one current holder: $\textsc{UnstaffedReserved} = \emptyset$.

Rationale. After a reorganisation a reserved decision may have no holder; the step is then neither automatable nor performable, and the process stops or is performed informally by an agent outside its charter—an incident **CC6** reports afterwards. Reserved work and oversight are the same scarce human resource; ReservedLoad is reported beside the oversight span.

Proposition 8.7 (**WF15**)**.** *D15 is evaluable in $O(|\textsc{Reserved}| + |\textsc{Performs}| + |\textsc{Holds}|)$ with indexing and creates no authority. When **WF2** and **WF15** both hold, every step that is agentic or reserved has a performer—an agent or a staffed role.*

Proof. Indexing; Corollary 8.2; the union of the two positive forms. □

8.14 The cap-composition profile and WF16

New in Charterion v1.1

WF6 bounds delegated *levels* by the meet; nothing bounded delegated *amounts*. A capped agent could delegate its whole cap to each of five delegatees, multiplying the exposure by five while every delegation looked bounded; **CC3** would see the resulting over-mandates only at the boundary, one at a time. The profile requires the quantitative profile of Section 8.6. Cumulative and per-period caps are out of scope for v1.1 (Appendix J).

Cap-composition profile: predicates and rules (stratum 5)

Extensional: $\textsc{DelegationCap}(a, b, \sigma, \lambda, u)$ — the permit cap carried by $\textsc{Delegation}(a, b, \sigma, \ell)$ in unit u.
$\textsc{OwnCap}(A, u) = \sum\{\lambda : \textsc{Charter}(M, A, \_, \_, \_),\ \textsc{Cap}(M, \lambda, \_, u)\}$ (defined when A has a capped charter in u); $\textsc{FanOut}(A, u) = \sum\{\lambda : \textsc{DelegationCap}(A, \_, \_, \lambda, u)\}$.
(D16a) $\textsc{CapAmplification}(A, u) \leftarrow \textsc{OwnCap}(A, u)$ defined, $\textsc{FanOut}(A, u) > \textsc{OwnCap}(A, u)$.
(D16b) $\textsc{UncappedDelegation}(A, B, \sigma) \leftarrow \textsc{Delegation}(A, B, \sigma, \ell),\ \ell \geq$ *commit*, $\textsc{OwnCap}(A, \_)$ defined, $\neg\exists \lambda, u\, \textsc{DelegationCap}(A, B, \sigma, \lambda, u)$.

Condition 8.9 (**WF16**: Bounded cap composition)**.** When the cap-composition profile is in use, the sum of the permit caps an agent delegates, per unit, SHALL not exceed the sum of the permit caps of its own charters in that unit, and every write-level delegation of a capped agent SHALL carry a cap.

Proposition 8.8 (**WF16**). *D16a–D16b are evaluable in $O(|\textsc{Cap}| + |\textsc{DelegationCap}| + |\textsc{Delegation}|)$ and create no authority. When **WF16** holds and **CC3** holds for every delegatee, the caps of the mandates issued by a's delegatees from those delegations total at most $\textsc{OwnCap}(a, u)$ per unit.*

Proof. Sums over indexed relations; Corollary 8.2; **CC3** bounds each delegatee's mandates by its caps, whose sum **WF16** bounds. The delegatees' own charters are not included, which is why the bound concerns the caps *from those delegations*. □

8.15 The obligation profile and WF17

New in Charterion v1.1

Reserved steps are business decisions; obligations are their legal grounding. The obligation profile records which instrument—a regulation article, a standard clause, an internal policy—imposes what kind of obligation on a scope, reports autonomous authority where an instrument requires a human decision, and makes the regulatory tables of Chapter 19 *computed* outputs (Appendix M).

Definition 8.10 (Obligation). $\textsc{Obligation}(o, \sigma, \mathit{instr}, \mathit{kind})$ records that instrument instr imposes on scope $\sigma \in \Sigma$ an obligation of $\mathit{kind} \in \{\text{human\_decision}, \text{record}, \text{oversight}, \text{information}\}$: *human_decision*—the decision at the steps of σ is taken by a natural person; *record*—decisions at σ are logged with their warrant; *oversight*—a natural person can monitor and halt agents acting at σ; *information*—affected persons are informed. Only *human_decision* yields a defect in v1.1; the other kinds ground computed evidence.

Obligation profile: rules (stratum 5)

(D17) $\textsc{UnmetObligation}(O, A, S) \leftarrow$
$\textsc{Obligation}(O, \sigma, \_, \text{human\_decision}),\ \textsc{InScope}(\sigma, S),\ \textsc{Auth}(A, S, \mathit{autonomous})$.
Metric (informative):
$\textsc{UngroundedReserved}(S) \leftarrow \textsc{Reserved}(S),\ \neg\exists O, \sigma\,(\textsc{Obligation}(O, \sigma, \_, \_) \wedge \textsc{InScope}(\sigma, S))$.

Condition 8.10 (**WF17**: Grounded obligation). When the obligation profile is in use, no agent SHALL hold autonomous authority on a step within the scope of a human-decision obligation: $\textsc{UnmetObligation} = \emptyset$.

Rationale. **WF17** catches what **WF8** cannot: a step that an instrument requires to be a human decision but that the business did not reserve, on which an agent is autonomous. No acceptance code is defined for it. The instrument labels are data supplied by the organisation; the framework computes evidence and does not interpret law (Chapter 19).

Proposition 8.9 (**WF17**). *D17 is evaluable in $O(\sum_o |\text{steps}(\sigma_o)| \cdot \bar{a})$, where $\bar{a}$ bounds the agents with authority on a step, and creates no authority. When **WF8** and **WF17** hold, no agent holds write-level authority on a reserved step in the scope of a human-decision obligation and none holds autonomous authority on any other step of that scope.*

Proof. Join against AUTH indexed by step; Corollary 8.2; conjunction of the positive forms. □

8.16 The perimeter profile and WF18

New in Charterion v1.1

Orchestrators learn tools and sub-agents at run time. **WF6** bounds the level of each delegation and **WF11** examines chains for separation; neither says which agents may delegate to which. The perimeter profile makes the delegation graph a designed object: declared collaboration edges, no cycles, a depth bound.

Perimeter profile: predicates and rules (stratum 5)

Extensional: $\textsc{Collaboration}(a, b)$ — a may delegate to b; parameter δ (depth bound, Conformance Statement). $\textsc{Del}^*$ is the closure of Definition 8.4.
(D18a)
$\textsc{OutOfPerimeter}(A, B) \leftarrow \textsc{Delegation}(A, B, \_, \_),\ \exists\textsc{Collaboration}(\_, \_),\ \neg\textsc{Collaboration}(A, B).$
(D18b) $\textsc{DelegationCycle}(A) \leftarrow \textsc{Del}^*(A, A).$
(D18c) $\textsc{DepthExceeded}(A, B) \leftarrow \textsc{Dist}(A, B) > \delta$, where $\textsc{Dist}$ is the length of the shortest delegation path.

Condition 8.11 (**WF18**: Delegation perimeter)**.** When the perimeter profile is in use, every delegation SHALL follow a declared collaboration edge, no agent SHALL delegate transitively to itself, and no delegation chain SHALL be longer than δ.

Rationale. Cycles are admissible in the v1.0 semantics ($\textsc{Del}^*$ is a closure) but never intended; the restated Proposition 8.3 handles them, and **WF18** reports them. A request that would create a delegation outside the perimeter is returned by the pre-analysis of the Authority Request Protocol (Chapter 16), so agents cannot extend the perimeter themselves.

Proposition 8.10 (**WF18**)**.** *D18a–D18c are evaluable in $O(|\textsc{Delegation}| + |\textsc{Collaboration}| + |A| \cdot |\textsc{Delegation}|)$ (one breadth-first search per delegator) and create no authority. With $\delta = 1$, every $\textsc{Dep}(a, b)$ of Definition 8.4 is a declared collaboration or a pair of declared collaborations from a common delegator.*

Proof. Breadth-first search; Corollary 8.2; with $\delta = 1$ every $\textsc{Del}^*$ pair is one delegation, hence a collaboration by D18a. □

8.17 The containment profile and WF19

New in Charterion v1.1

Reach and blast radius were metrics in v1.0 (Section 8.7, Chapter 20); the containment profile makes them conditions. The bound is the architecture's statement of how much a compromise of an agent may cost; Appendix N states the adversarial assumptions under which the bound holds.

Containment profile: predicates and rules (stratum 5)

Extensional: $\textsc{BlastBound}(a, n)$; $\textsc{AgentClass}(a, c)$; $\textsc{ClassBlastBound}(c, n)$. $\textsc{Blast}$ is that of Definition 8.7.
$\textsc{Bound}(A, n) \leftarrow \textsc{BlastBound}(A, n).$
$\textsc{Bound}(A, n) \leftarrow \textsc{AgentClass}(A, C),\ \textsc{ClassBlastBound}(C, n),\ \neg\exists n'\ \textsc{BlastBound}(A, n').$
(D19) $\textsc{BlastExceeded}(A) \leftarrow \textsc{Bound}(A, n),\ |\textsc{Blast}(A)| > n.$

Condition 8.12 (**WF19**: Bounded blast radius)**.** When the containment profile is in use, no agent's blast radius SHALL exceed its declared bound, own or class: $\textsc{BlastExceeded} = \emptyset$.

Proposition 8.11 (**WF19**)**.** *D19 is evaluable in $O(|A|)$ given $\textsc{Blast}$ and creates no authority. Under **WF19** and the assumptions of Appendix N, the resources a compromised agent a can write through its own authority or any chain of its delegations number at most $\textsc{Bound}(a)$.*

Proof. Lookup; Corollary 8.2; Definition 8.7. □

Table 8.3. The v1.1 extension profiles: conditions, defects and profiles (all optional; recommended SHOULD at the levels of Table 25.2)

Id	Name	Defect predicates	Profile	Reads (strata ≤ 4)
WF13	Configuration currency	ConfigDrift(a, m), Unreviewed(m)	configuration	Charter (M_t)
WF14	Chartered instantiation	Uncharted(i), OverInstantiated(a), OverloadedI(h)	instance	Charter, Delegation, Oversight, Escalation
WF15	Staffed reservation	UnstaffedReserved(s)	staffing	Reserved, Holds
WF16	Bounded cap composition	CapAmplification(a, u), UncappedDelegation(a, b, σ)	cap-composition (needs quantitative)	Cap, Delegation
WF17	Grounded obligation	UnmetObligation(o, a, s)	obligation	InScope, Auth
WF18	Delegation perimeter	OutOfPerimeter(a, b), DelegationCycle(a), DepthExceeded(a, b)	perimeter	Delegation, Del*
WF19	Bounded blast radius	BlastExceeded(a)	containment	Blast
CC7	Certified envelope (Section 11.5)	Uncertified(e), InvalidCertificate(e)	certificate	Envelope Register, EDB

9 The execution plane: the substrate

The execution plane is the Substrate Framework of the third article (El Hassani, 2026c), reproduced here in its normative form. Its unit of organisation is not the element but the *assertion*: a statement about an element or a relation that says where it came from, how confident the organisation is in it, for how long it holds, and whether a human has stood behind it. This is what makes the model usable by an actor that cannot interpolate: the model can say what it does not know.

9.1 Assertions and the substrate

Definition 9.1 (Warrant-bearing assertion)**.** An *assertion* is a tuple $a = \langle s, p, o, \sigma, \kappa, \tau, \varepsilon \rangle$ where s is the subject (an identifier), p a predicate drawn from the vocabulary of one layer, o an object (an identifier or a literal), σ the provenance (the source that produced the assertion and, if approved, the approver), $\kappa \in [0, 1]$ the confidence, $\tau = [t_{\text{from}}, t_{\text{until}})$ the validity interval, and $\varepsilon \in \{observed, approved, derived\}$ the epistemic status (a fourth status, *attested*, is added at the boundary, Chapter 10). An assertion is *current* at time t if $t \in \tau$; it is *authoritative* if $\varepsilon = approved$.

Definition 9.2 (Substrate)**.** A *substrate* is a tuple $\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{G}, \mathcal{L} \rangle$ where $\mathcal{A}$ is a set of assertions partitioned by layer into $\mathcal{A}_{\text{anc}}, \mathcal{A}_{\text{aff}}, \mathcal{A}_{\text{env}}, \mathcal{A}_{\text{coord}}$, $\mathcal{H}$ the set of human principals entitled to approve, $\mathcal{G}$ the set of agent principals, and $\mathcal{L}$ the decision log. The *warrant layer* is not a partition of $\mathcal{A}$ but the attributes $\langle \sigma, \kappa, \tau, \varepsilon \rangle$ carried by every assertion together with $\mathcal{L}$.

Observed content is produced by any source, including agents that mine the estate; it can be stored freely and is the mechanism by which agents maintain the model. *Approved* content has been validated by a human in $\mathcal{H}$ and is the only content that can authorise an action. *Derived* content is computed from other assertions by a stated rule and inherits the minimum confidence and the intersection of validity intervals of its premises. Confidence and validity are the two components of warrant: confidence expresses how sure the source is that the assertion was true when made; validity expresses how long the organisation is prepared to rely on it without re-observation—a commitment about acceptable staleness, not a prediction of change.

9.2 The five layers

Table 9.1 gives the Substrate Grid: for each layer, the actor's question it answers, its constructs, the type of answer it returns, its typical sources of authority and its approval regime. The anchor layer establishes identity and typing; the affordance layer binds operations to elements and interfaces and states preconditions, effects, reversibility and side-effect scope—it is the join between the structural description of an application interface and the invocability of a tool description; the envelope layer states, for a principal, the scope of its authority and the rules that evaluate a request to permit, deny or escalate; the coordination layer records shared referents, declared intents, commitments, locks and ownership, so that a second agent or a human can see what a first agent is about to change; the warrant layer is cross-cutting.

Table 9.1. The Substrate Grid (El Hassani, 2026c)

Layer	Question of the actor	Core constructs	Answer type	Typical sources of authority	Approval regime
Anchor	Does it exist, and is it the one I mean?	Entity, identifier, type, name, synonym, part-of, realizes	Identifier or unresolved	CMDB, master data, code repositories, EA repository	Observed by default; approved for identity merges and retirements
Affordance	Can it be done, through what, with what consequence?	Operation, interface, precondition, effect, side-effect scope, reversibility	Operation binding or unavailable	API catalogues, OpenAPI/MCP descriptions, runbooks	Observed; approved for effect and reversibility claims
Envelope	May I, and where must I stop?	Authority scope, rule, threshold, SoD constraint, escalation condition, principal	permit, deny or escalate	Architecture board, risk and security functions, regulation; *in Charterion: derived from charters*	Approved only; never observed-only (**Inv-2**)
Coordination	With whom or what do I share the estate?	Shared referent, commitment, intent, lock, ownership, accountable unit	Referent and current commitments	Process frames (Agentic BPM), ownership registries, ticketing	Observed for intents; approved for ownership
Warrant	How far can I trust the answer?	Provenance σ, confidence κ, validity τ, status ε, approver, decision log	Minimum warrant of supporting assertions, or unwarranted	Every source, recorded per assertion	Attributes of every assertion; approver mandatory for $\varepsilon =$ approved

9.3 The admissibility function

Definition 9.3 (Request, admissibility)**.** A *request* is $q = \langle g, op, x, c, t\rangle$: agent g proposes operation op on the target named x in context c at time t. The *admissibility function* $\mathrm{ADM} : Q \times \mathcal{S} \to \{permit, deny, escalate, unwarranted\} \times 2^{\mathcal{A}}$ is defined by Algorithm 1 with two parameters: θ_{id}, the minimum confidence for identity resolution, and θ_{act}, the minimum warrant for acting without a human.

The four outcomes have distinct remedies, which is why they are not collapsed. *permit* lets the agent act and, in the lifecycle, re-observe. *deny* stops the agent; no human can lift it at run time, because it expresses an unsatisfied precondition or an approved rule such as a separation-of-duty constraint, and lifting it is a change to the substrate, not a decision about the request. *escalate* means the approved envelope permits the action subject to a human decision—a threshold has been crossed or a human-in-the-loop condition applies—and that decision is taken by the escalation human of Definition 8.2. *unwarranted* means the substrate cannot stand behind an answer; its remedy is re-observation or a human answer that becomes a new assertion. The survey's two most preferred oversight modes—validation above model-defined thresholds and validation only where the model declares it cannot warrant the action—are exactly *escalate* and *unwarranted*.

Proposition 9.1 (Properties of admissibility)**.** (*P1, deny dominance*) *If any approved envelope rule evaluates to deny, the outcome is deny regardless of warrant.* (*P2, approved-only authorisation*) *The outcome is permit or escalate only if at least one approved envelope assertion for g is current; observed or derived envelope content never contributes to V.* (*P3, monotonicity in warrant*) *For fixed envelope and premises, lowering the confidence or shortening the validity of any supporting assertion can change the outcome only towards unwarranted, never from deny to permit or from escalate to permit.*

Algorithm 1 Admissibility ADM$(q, \mathcal{S})$ for $q = \langle g, op, x, c, t \rangle$ (El Hassani, 2026c)

$W \leftarrow \emptyset$ ▷ supporting assertions
$e \leftarrow \mathrm{resolve}(x, t, \theta_{\mathrm{id}})$ ▷ anchor: current identity assertions with $\kappa \geq \theta_{\mathrm{id}}$
if e is none or ambiguous **then**
  return $(\mathit{unwarranted}, W)$
end if
$W \leftarrow W \cup \{\text{identity assertions of } e\}$
$F \leftarrow \{a \in \mathcal{A}_{\mathrm{aff}} : a.s = e,\ a.p = \mathit{affords},\ a.o = op,\ t \in a.\tau\}$ ▷ affordance
if $F = \emptyset$ **then**
  return $(\mathit{unwarranted}, W)$ if an expired affordance exists, else (deny, W)
end if
$W \leftarrow W \cup F$
for all preconditions (p', v) of op on e **do**
  $P \leftarrow \{a \in \mathcal{A} : a.s = e,\ a.p = p',\ t \in a.\tau\}$
  if $P = \emptyset$ **then**
    return $(\mathit{unwarranted}, W)$
  end if
  $W \leftarrow W \cup P$
  if $\mathrm{latest}(P).o \neq v$ **then**
    return (deny, W)
  end if
end for
$E \leftarrow \{a \in \mathcal{A}_{\mathrm{env}} : a.s = g,\ t \in a.\tau,\ a.\varepsilon = \mathit{approved}\}$ ▷ envelope: approved only (**Inv-2**)
if $E = \emptyset$ **then**
  return $(\mathit{unwarranted}, W)$
end if
$V \leftarrow \{\mathrm{eval}(a, q, e) : a \in E\}$; $W \leftarrow W \cup E$
if $\mathit{deny} \in V$ **then**
  return (deny, W)
end if
$\kappa_{\min} \leftarrow \min_{a \in W} a.\kappa$ ▷ warrant
if $\kappa_{\min} < \theta_{\mathrm{act}}$ **then**
  return $(\mathit{unwarranted}, W)$
end if
if $\mathit{escalate} \in V$ **then**
  return $(\mathit{escalate}, W)$
end if
if $\mathit{permit} \notin V$ **then**
  return $(\mathit{unwarranted}, W)$
end if
return (permit, W)

Proof. P1 and P2 are immediate from the deny test and the construction of E; P3 holds because $\kappa_{\min}$ and currency enter only through tests whose failure returns unwarranted. □

New in Charterion v1.1

Positioning of the four-valued function (informative). Decision outcomes beyond *permit* and *deny* are not an invention of this framework. XACML 3.0 (OASIS Standard, 2013) returns Permit, Deny, Indeterminate and NotApplicable, with extended Indeterminate values and a `missing-attribute` status; AARM (2026) returns allow, deny, modify, step up and defer; the OWASP Agent Control Standard (2026) returns `allow`, `deny`, `modify`, `ask` and `defer`; the OpenID AuthZEN Authorization API 1.0 (2026) is binary. What is specific to Definition 9.3 is not the arity but three commitments: *escalate* is routed to a human named in the design model (Section 8.2), not to "the user"; *unwarranted* is grounded in the warrant of the assertions consulted and is kept distinct from *escalate* (which defer and `ask` conflate) and from *deny*; and the outcome is monotone in the warrant (property P3), which none of the four states. Primary texts retrieved 6 September 2026; Section 24.5.

9.4 The lifecycle

A substrate that agents query at the moment of acting cannot be maintained by projects; it is maintained continuously, by the agents it governs and by the humans who approve what they may do. The Substrate Lifecycle is a loop of six phases, which Part III embeds as the operation loop of the method: **Observe** (systems, agents and humans emit observed assertions with source and validity; agents re-observe the slice they acted upon), **Reconcile** (names are resolved, conflicts between federated sources are resolved by a per-domain precedence rule or left as two assertions with reduced confidence, derived assertions are recomputed), **Commit** (human principals approve changes to envelope and ownership assertions; nothing else requires approval), **Serve** (the admissibility function answers requests), **Account** (every decision is logged with its supporting assertions and minimum warrant, so that any action can be replayed against the substrate as it was), and **Detect drift** (outcomes are compared with what the estate later shows, and validity intervals and confidences are re-estimated). There is no "target architecture" phase: the substrate describes the estate as it is and the authority as it stands, and transformation work becomes one producer of approved assertions among others.

9.5 Invariants

Invariants of the execution plane (El Hassani, 2026c)

Inv-1 (Human approval)
An assertion with $\varepsilon = \mathit{approved}$ records an approver in $\mathcal{H}$. An agent cannot approve.

Inv-2 (Separation of observed and approved)
Envelope assertions authorise only when approved. Observed envelope content may be stored, for example as an agent's proposal, but the admissibility function does not read it.

Inv-3 (Non-self-authorisation)
No envelope assertion whose subject is agent g may have g as its source, whatever its status, and no agent may be the approver of any assertion. An agent may propose a change to its own envelope only as an observed assertion whose subject is the proposal, not the rule.

Inv-4 (Declared boundary)
Every assertion has a finite validity interval, and the admissibility function returns unwarranted rather than a default when a premise is absent, expired, ambiguous or below θ_{act}.

Inv-5 (Accountability)
Every decision is logged with its supporting assertions, and the log is append-only and readable by the auditor.

Inv-1–**Inv-3** are enforced at write time by the substrate; **Inv-4** and **Inv-5** are properties of the admissibility function and the serving infrastructure. In Charterion the invariants are also governance gates (Section 18.6).

9.6 Conformance levels of the execution plane

Five cumulative levels give the path from a documentation-oriented EA practice to a warrant-bearing substrate. **Level 0** is documentation: no layer is machine-readable. **Level 1** makes the anchor layer queryable and supports read-only agents; identity resolution alone removes the misidentification failure that retrieval grounding cannot address. **Level 2** adds the affordance layer and supports reversible actions on elements whose operations and preconditions are known. **Level 3** adds an approved envelope and the full four-valued admissibility function, and supports irreversible actions under model-bounded oversight. **Level 4** adds drift detection and per-assertion re-estimation of validity, and permits agents to maintain the substrate under **Inv-1**–**Inv-3**. The level of a substrate is the level of its least conformant layer in the slice being acted upon, so that an organisation may run at Level 3 for payments and at Level 1 elsewhere. Chapter 25 combines these with the design-plane and boundary levels into the Charterion maturity model.

9.7 From charters to envelopes

New in Charterion v1.0

The relation between the agency layer and the envelope layer is stated here for the first time as a construction. In the articles the two planes shared only the action-level vocabulary; here the design plane *produces* the envelope.

Definition 9.4 (Level of an operation, envelope entry)**.** Every operation op of the affordance layer is assigned a level $\mathrm{level}(op) \in \mathbb{L}$ by its effect and reversibility assertions: an operation without effect is *read*; an operation whose effect is the creation of a proposal, draft or reservation that another principal must confirm is *propose*; an operation with a state-changing effect and a declared reversibility or confirmation procedure is *commit*; an operation with a state-changing effect and neither is *autonomous*. An *envelope entry* for agent g is an approved envelope assertion $\textsc{authorityScope}(g) \ni \langle x, \ell \rangle$ together with the rules that apply to it; it *permits* $\langle g, op, x, \cdot, t \rangle$ when current and $\mathrm{level}(op) \le \ell$.

Definition 9.5 (Charter-derived envelope)**.** The *charter-derived envelope* of agent g at instant t is the set of envelope entries $\langle x, \ell \rangle$ such that $\textsc{Need}(g, x, \ell)$ holds in the least model of M_t, each with provenance σ = the charter identifiers from which the need derives, approver set $\mathrm{Acc}(g, \cdot)$ restricted to $\mathcal{H}$, validity equal to the intersection of the validity intervals of those charters, and the following rules: a threshold rule from $\textsc{Cap}$ when the quantitative profile is in use (permit below $\lambda_{\mathrm{permit}}$, escalate up to $\lambda_{\mathrm{escalate}}$, deny above); a separation-of-duty rule for every $\mathrm{SoD}(s, s')$ with s covered by g; and an escalation condition naming $\mathrm{Esc}(g, \cdot)$ for every entry at $\ell \ge$ *commit* whose charter is $\textsc{Stale}$, and for every entry at *autonomous* (the human of $\textsc{Oversight}$ may halt).

The construction is what bridge B1 of the method performs (Chapter 12), and **CC1**–**CC2** of Chapter 11 are the conditions that the envelope actually in force equals the charter-derived one. Two consequences follow immediately. First, every envelope entry has a human approver by Proposition 7.3, so **Inv-1** is satisfied by construction for derived entries. Second, the derived envelope is an *approved* assertion in the sense of **Inv-2** only once the approvers have committed it in phase **Commit**; the design plane proposes, the humans it names approve, and the substrate serves—which is the separation that 83 % of surveyed practitioners required.

10 The boundary plane

The boundary plane is the Boundary Substrate of the fourth article (El Hassani, 2026d). It addresses the situation in which an agent of organisation A acts on, or transacts with, an agent of organisation B, and neither organisation's approvers can approve the other's content. Four things differ from the intra-organisational case: authority must travel (the acting agent presents an authority conferred at home); the target must be declared (the receiving organisation decides what it lets foreign agents see and do); warrant is asymmetric (each side weighs the other's claims by a trust it alone controls); and outcomes are joint (an action admissible on one side and not on the other is not admissible).

10.1 Constructs

Definition 10.1 (Mandate)**.** A *mandate* is a tuple $\mu = \langle A, g, \Sigma_\mu, C_\mu, \lambda_{\text{permit}}, \lambda_{\text{escalate}}, \tau, h \rangle$ issued by organisation A to its agent g: Σ_μ is the set of operations (with their targets) the agent may perform abroad, C_μ the set of counterparties with which it may perform them, λ_{permit} and $\lambda_{\text{escalate}}$ the quantitative bands in a stated unit, τ the validity interval and $h \in \mathcal{H}_A$ the human who issued it. A mandate is a portable, signed form of an approved envelope assertion; it is presented to the counterparty with every request.

Definition 10.2 (Exposure)**.** The *exposure* X_B of organisation B is the part of its substrate it declares to foreign agents: an *external affordance inventory* (the operations, targets, preconditions and effects available across the boundary) and a *boundary envelope* (approved rules stating which counterparties, under which mandates and within which bands, may invoke each operation, and which invocations escalate to a human of B). The exposure is approved content of B under **Inv-1**–**Inv-3**; no foreign principal contributes to it.

Definition 10.3 (Attested status, trust weight, effective confidence)**.** Content received from a counterparty—its registry identity, its catalogue, its mandates, its commitments—enters the receiving substrate with the epistemic status $\varepsilon = \mathit{attested}$: neither observed by A nor approved by A's humans, but asserted by B under B's authority. Each organisation maintains a *trust weight* $t_A(B) \in [0, 1]$ for each counterparty, set by its own approvers and re-estimated from the commitment record. The *effective confidence* of an attested assertion with stated confidence κ is $\kappa_{\text{eff}} = \kappa \cdot t_A(B)$, and it is κ_{eff} that the admissibility function uses.

Definition 10.4 (Reciprocal commitment)**.** A *commitment* $\mathcal{C}(\mathit{debtor}, \mathit{creditor}, \mathit{antecedent}, \mathit{consequent})$ in the sense of Singh (1999) records that the debtor organisation is committed to the creditor to bring about the consequent once the antecedent holds. Every cross-boundary action that changes state on either side creates a commitment recorded, with a shared referent, in the coordination layer of *both* substrates, and its lifecycle (active, satisfied, violated, cancelled) is the evidence from which trust weights are re-estimated.

10.2 Two-sided admissibility

Definition 10.5 (Two-sided admissibility)**.** For a cross-boundary request q by agent g of A on exposure X_B under mandate μ,

$$\mathrm{ADM}^{\times}(q) = \max_{\preceq}\big(\mathrm{ADM}_{\mathrm{out}}(q, \mathcal{S}_A, \mu),\ \mathrm{ADM}_{\mathrm{in}}(q, X_B, \mu)\big),$$
$$\mathit{permit} \prec \mathit{escalate} \prec \mathit{unwarranted} \prec \mathit{deny},$$

where $\mathrm{ADM}_{\mathrm{out}}$ evaluates the request against A's own envelope (is g mandated to do this, here, with this counterparty, within these bands?) and $\mathrm{ADM}_{\mathrm{in}}$ evaluates it against B's boundary envelope and affordance inventory with attested content weighted by $t_B(A)$. An *escalate* outcome is decided by a human of the side that produced it; if both sides escalate, each decides its own.

Invariants of the boundary plane (El Hassani, 2026d)

X-Inv-1 (Human issuance)
A mandate is issued by a human of the issuing organisation; no agent issues or extends a mandate, and a mandate carried by a delegatee is bounded by the mandate of the delegator.

X-Inv-2 (Trust locality)
The trust weight $t_A(B)$ is set and re-estimated only by A's approvers from A's own commitment record; no counterparty, registry or intermediary writes it.

Proposition 10.1 (Properties of the boundary plane)**.** (*X-P1, attenuation*) *An agent's authority abroad under a mandate never exceeds its authority at home:* $\mathrm{ADM}^{\times}(q) \succeq \mathrm{ADM}_{\mathrm{out}}(q)$. (*X-P2, non-amplification along chains*) *Under* ***X-Inv-1****, along any chain of mandates and delegations the bands and operation sets are non-increasing.* (*X-P3, trust monotonicity*) *Lowering* $t_A(B)$ *can change* $\mathrm{ADM}_{\mathrm{in}}$ *only towards unwarranted* (*by P3 of Proposition 9.1 applied to* κ_{eff}). (*X-P4, approver locality*) *Every escalate outcome is decided by a human of the organisation whose envelope produced it, and no human of either organisation approves content of the other.*

Proof. X-P1 is the definition of the maximum. X-P2 follows from **X-Inv-1** by induction as in Proposition 7.2. X-P3 is P3 with κ replaced by $\kappa \cdot t_A(B)$, monotone in $t_A(B)$. X-P4 is the definition of $\mathrm{ADM}^{\times}$ together with **Inv-1** applied to each substrate. □

10.3 Exposure levels

Five levels give the path from a closed organisation to one that settles business with foreign agents. **X0 Closed**: no exposure; foreign agents are refused at the boundary. **X1 Discoverable**: an attested catalogue (anchor and affordance inventory) is exposed read-only, so a foreign agent can find out what exists and what could be requested. **X2 Reservable**: foreign agents may perform reversible, non-binding operations (quote, reserve, hold) under mandates; every operation has a declared reversal. **X3 Committing**: foreign agents may perform binding operations (order, accept, amend) under mandates within bands; commitments are recorded on both sides; escalations are decided locally. **X4 Settling**: foreign agents may trigger settlement (payment, transfer of title) within bands; trust weights are re-estimated from the commitment record on a declared cadence. An exposure level is declared per relationship, and **CC4** (Chapter 11) bounds it by the internal conformance levels of both sides: an organisation cannot let foreign agents commit against a slice whose own substrate cannot warrant the action.

10.4 From charters to mandates

New in Charterion v1.0

As with envelopes, the mandate is derived from the design plane. A mandate for agent g is issued by a human $h \in \mathrm{Acc}(g, \cdot)$ (**CC5**); its operation set Σ_μ is drawn from operations whose level is covered by $\textsc{Need}(g, \cdot, \cdot)$ on the resources those operations touch (**CC3**); its bands are bounded by $\textsc{Cap}$ of the corresponding charter when the quantitative profile is in use; and its validity is bounded by the charter's. The Mandate Register (Chapter 21) records the charter identifiers from which each mandate derives. Under these conditions X-P1 extends to the design plane: an agent holds no more authority abroad than its charters justify at home.

11 Cross-plane coherence

New in Charterion v1.0

This chapter is the principal formal addition of v1.0. It states six conditions under which the three planes describe one and the same authority, proves that under them every action an agent is permitted to perform at run time, at home or abroad, is traceable to a charter and to a human accountable for the agent, and shows that each condition is evaluable in polynomial time from the model, the envelope and the mandate and decision registers.

11.1 Setting

Let M be a design-plane model and M_t its time-indexed form at instant t (Definition 8.3). Let $\mathcal{E}_t$ be the set of envelope entries of Definition 9.4 approved and current at t, written $\langle g, x, \ell, appr\rangle$ with $appr$ the approver. Let $\mathcal{M}_t$ be the set of mandates current at t and $\mathcal{L}$ the decision log, whose records are $\langle q, outcome, W, time\rangle$. Let $\mathrm{level}(op)$ be as in Definition 9.4, and let $\mathrm{res}(op)$ be the resource of the target of op in the anchor layer. All conditions are stated as defect predicates; the condition holds when the predicate is empty.

11.2 The six conditions

Condition 11.1 (**CC1**: Grounded envelope)**.** Every approved envelope entry is justified by a derived need at or above its level:

$$\textsc{UngroundedEnvelope}(g, x, \ell) \leftarrow \langle g, x, \ell, \_\rangle \in \mathcal{E}_t,\ \neg\textsc{Need}^{M_t}_{\geq}(g, x, \ell).$$

Reading: no run-time authority without a design-time charter (axiom X1). An envelope entry added by a platform team by hand, or left behind by a charter since narrowed, expired or degraded, is reported.

Condition 11.2 (**CC2**: Complete envelope)**.** Every derived need is served by a current approved envelope entry at or above its level:

$$\textsc{UnservedNeed}(g, x, \ell) \leftarrow \textsc{Need}^{M_t}(g, x, \ell),\ \neg\exists\, \ell' \geq \ell : \langle g, x, \ell', \_\rangle \in \mathcal{E}_t.$$

Reading: the run-time dual of **WF3**-missing. The architecture has chartered the agent to do something the substrate will deny or leave unwarranted; the agent will fail in production, and the failure is visible before it happens. **CC1** and **CC2** together state that the envelope in force *is* the charter-derived envelope of Definition 9.5.

Condition 11.3 (**CC3**: Bounded mandate)**.** Every mandate is within the needs and caps of the agent's charters:

$$\textsc{OverMandate}(\mu) \leftarrow \mu \in \mathcal{M}_t,\ op \in \Sigma_\mu,\ \neg\textsc{Need}^{M_t}_{\geq}(g_\mu, \mathrm{res}(op), \mathrm{level}(op)).$$

$$\textsc{OverMandate}(\mu) \leftarrow \mu \in \mathcal{M}_t,\ \textsc{Cap}(m, \lambda_p, \lambda_e, u),\ m \text{ grounds } \mu,\ (\lambda^{\mu}_{\mathrm{permit}} > \lambda_p \ \vee\ \lambda^{\mu}_{\mathrm{escalate}} > \lambda_e).$$

Reading: no authority abroad beyond authority at home (axiom X5). The second rule applies only when the quantitative profile is in use and the mandate names the charter(s) that ground it in the Mandate Register.

Condition 11.4 (**CC4**: Level alignment)**.** For every slice ς of the organisation with execution-plane conformance level $L(\varsigma)$ and for every relationship ρ with exposure level $X(\rho)$ that touches ς:

$$\textsc{LevelMismatch}(\varsigma) \leftarrow \exists\, a, s \in \varsigma :\ \textsc{Auth}^{M_t}(a, s, \ell),\ \ell \geq \mathit{commit},\ L(\varsigma) < 3.$$
$$\textsc{LevelMismatch}(\varsigma) \leftarrow X(\rho) \geq X2 \wedge L(\varsigma) < 2; \quad X(\rho) \geq X3 \wedge L(\varsigma) < 3; \quad X(\rho) = X4 \wedge L(\varsigma) < 4.$$

Reading: no irreversible autonomy on a slice the substrate cannot warrant. A charter that confers write-level authority on steps of a slice whose substrate has no approved envelope (Level < 3) is an authority nothing enforces; an exposure that lets foreign agents commit (X3) against such a slice is worse. **CC4** is evaluated from the Conformance Statement, which records L per slice and X per relationship.

Condition 11.5 (**CC5**: Accountability continuity)**.** The approver of every envelope entry and the issuer of every mandate for agent g is a human accountable for g in the design plane:

$$\textsc{BrokenChain}(g, h) \leftarrow \langle g, \_, \_, h\rangle \in \mathcal{E}_t,\ \neg\textsc{Acc}^{M_t}(g, h).$$
$$\textsc{BrokenChain}(g, h) \leftarrow \mu \in \mathcal{M}_t,\ g_\mu = g,\ h_\mu = h,\ \neg\textsc{Acc}^{M_t}(g, h).$$

Reading: the human who stands behind the run-time rule is the one the architecture names (axiom X2). **CC5** is what makes **Inv-1** and **X-Inv-1** *meaningful* rather than merely satisfied: an approver who is a human but not an accountable one—a platform administrator, a departed manager whose account survives—is reported.

Condition 11.6 (**CC6**: Evidence return)**.** Every denied, escalated, unwarranted or later-reversed decision in the log is attributable to at least one charter of the requesting agent, and is counted in that charter's evidence record:

$$\textsc{UnattributedEvidence}(r) \leftarrow r = \langle\langle g, op, x, \_, t'\rangle, o, \_, \_\rangle \in \mathcal{L},\ (o \neq \mathit{permit} \ \vee\ \mathrm{reversed}(r)),$$
$$\neg\exists\, m, \sigma, s :\ \textsc{Charter}(m, g, \sigma, \_, \_),\ \textsc{InScope}(\sigma, s),\ \textsc{Requires}(s, x, \_).$$

Reading: the loop closes. A decision that no charter explains is either an agent acting outside its architecture (an incident) or a charter missing from the model (a modelling defect); both must be found. Records that *are* attributed feed the evidence record of the charter and the evidence-gated promotion of Chapter 18.

11.3 End-to-end properties

Theorem 11.1 (End-to-end accountability)**.** *Let M, $\mathcal{E}_t$ and $\mathcal{S}$ satisfy **Inv-1–Inv-3**, **CC1** and **CC5** at instant t. If $\mathit{ADM}(\langle g, op, x, c, t\rangle, \mathcal{S}) = \mathit{permit}$ or $\mathit{escalate}$, then there exist a charter $\textsc{Charter}(m, a_0, \sigma, \ell_0, \pi)$, a (possibly empty) delegation chain from a_0 to g, and a human h such that (i) $\textsc{Need}^{M_t}(g, x, \ell)$ with $\ell \geq \mathrm{level}(op)$ derives from m through that chain, (ii) $\mathit{Acc}^{M_t}(g, h)$ and $\mathit{Acc}^{M_t}(a_0, h)$, and (iii) the approver of the envelope entry that produced the outcome is such an h.*

Proof. By P2 (Proposition 9.1) a permit or escalate outcome requires a current approved envelope entry $\langle g, x, \ell', \mathit{appr}\rangle \in \mathcal{E}_t$ that evaluated to permit or escalate for op, so $\ell' \geq \mathrm{level}(op)$ by Definition 9.4. By **CC1**, $\textsc{Need}^{M_t}_{\geq}(g, x, \ell')$, hence $\textsc{Need}^{M_t}(g, x, \ell)$ for some $\ell \geq \ell'$ by (N2). By (N1) there is a step s with $\textsc{Covers}(g, s)$ and $\textsc{Requires}(s, x, \ell)$; by (C2) and (A1)–(A2), $\textsc{Auth}(g, s, \cdot)$ derives from some charter m of some a_0 through a delegation chain to g, which gives (i). By

CC5, $appr$ satisfies $\mathrm{Acc}^{M_t}(g, appr)$; by (K1)–(K3) every h with $\mathrm{Acc}(g, h)$ derived along the chain satisfies $\mathrm{Acc}(a_0, h)$, and if g's accountability derives from a charter of g itself then $a_0 = g$; this gives (ii) and (iii). **Inv-1**–**Inv-3** guarantee that $appr$ is a human and not g. □

Theorem 11.2 (End-to-end attenuation at the boundary)**.** *Under **X-Inv-1**, **CC3** and **CC5**, if $ADM^{\times}(q) \in \{permit, escalate\}$ for a request $q = \langle g, op, x, c, t\rangle$ under mandate μ, then $\textsc{Need}^{M_t}_{\geq}\big(g, \mathrm{res}(op), \mathrm{level}(op)\big)$ holds in the issuing organisation's design plane, the issuer h_μ is accountable for g, and, when the quantitative profile is in use, the amount of the action is below the escalate cap of a charter that grounds μ.*

Proof. By X-P1 the outcome is permit or escalate only if $\mathrm{ADM}_{\mathrm{out}}$ is, which requires $op \in \Sigma_\mu$ and the amount within μ's bands; **CC3** then gives the need and the cap bound; **CC5** gives accountability of h_μ. □

Proposition 11.1 (Complexity of coherence checking)**.** *Given the least model of M_t, **CC1**, **CC2**, **CC3** and **CC5** are evaluable in time $O(|\mathcal{E}_t| + |\textsc{Need}| + \sum_\mu |\Sigma_\mu|)$ with hash-indexed lookups, **CC4** in time linear in the number of slices and relationships, and **CC6** in time $O(|\mathcal{L}| \cdot c)$ where c bounds the number of charters per agent. Together with Proposition 7.4, the complete Charterion analysis is polynomial in the size of the model and the registers.*

Proof. Each rule is a join of one register against a derived relation of the least model on a key; with the derived relations indexed, each register row is examined once against a constant number of lookups. □

11.4 What coherence adds

The articles established, separately, that the design plane can be analysed (Propositions 7.1–7.4), that the execution plane can decide (Proposition 9.1), and that the boundary plane can compose decisions (Proposition 10.1). None of them could state that the authority decided at run time is the authority the architecture justified, because the two were connected by a shared vocabulary and by prose. The six conditions make the connection a property of data that a repository holds and a validator checks: the envelope is derived from charters and nothing else (**CC1**–**CC2**), mandates are bounded by charters (**CC3**), autonomy is granted only where the substrate can enforce it (**CC4**), the humans behind run-time rules are the humans the architecture names (**CC5**), and run-time evidence returns to the charters that caused it (**CC6**). Table 11.1 summarises the conditions, their sources of data and the axiom each operationalises.

Table 11.1. Cross-plane coherence conditions

Id	Name	Data compared	Defect	Axiom
CC1	Grounded envelope	Envelope entries vs. derived Need	UngroundedEnvelope	X1 no authority without charter
CC2	Complete envelope	Derived Need vs. envelope entries	UnservedNeed	X3 no entitlement without need (dual)
CC3	Bounded mandate	Mandate operations and bands vs. Need and Cap	OverMandate	X5 no mandate beyond charter
CC4	Level alignment	Write-level Auth and exposure levels vs. execution-plane level per slice	LevelMismatch	X6 no answer without warrant
CC5	Accountability continuity	Approvers and issuers vs. Acc	BrokenChain	X2 no charter without principal
CC6	Evidence return	Decision log vs. charters	UnattributedEvidence	X4 no autonomy without oversight (evidence side)

11.5 Authority certificates and CC7

New in Charterion v1.1

Theorem 11.1 says that behind every permitted or escalated decision there *exist* a charter, a delegation chain, a step, a need and an accountable human. In v1.0 an auditor who wanted to see them had to re-run the reasoner on the repository and trust both. Version 1.1 makes the derivation an object: an *authority certificate* carried by every envelope entry and every decision-log record, checkable by fact lookups against a model snapshot identified by its digest, without the reasoner and without trusting the repository that served it. The analogue in program verification is proof-carrying code, in which the producer ships a proof that a lightweight checker validates; the difference is that a Charterion certificate proves a *derivation of authority*, not a safety property of code. The certificate profile is optional; when it is in use, **CC7** requires the certificates to be present and valid.

Definition 11.1 (Authority certificate)**.** Let M be a model with EDB digest d. An *authority certificate* for the claim $\textsc{Need}_{\geq}(g, x, \ell)$ is the tuple

$$c = \big\langle\, \langle g, x, \ell\rangle;\ \langle m, a_0, \sigma, \ell_0, \pi\rangle;\ \langle (a_0, a_1, \sigma_1, \ell_1), \ldots, (a_{k-1}, a_k, \sigma_k, \ell_k)\rangle;\ s;\ w_0, \ldots, w_k;\ \mathrm{Req}(s);\ \ell^*;\ \langle \pi, H\rangle;\ d;\ \mathrm{digest}$$

with $a_k = g$, where $\langle m, a_0, \sigma, \ell_0, \pi\rangle$ is a charter fact, each $(a_{i-1}, a_i, \sigma_i, \ell_i)$ is a delegation fact, s is a step, each w_i is a path $[\sigma_i, \ldots, s]$ over Realizes and Step facts witnessing $\textsc{InScope}(\sigma_i, s)$ $(\sigma_0 = \sigma)$, $\mathrm{Req}(s) = \{(x', \ell') : \textsc{Requires}(s, x', \ell')\}$, $\ell^* = \ell_0 \sqcap \ell_1 \sqcap \ldots \sqcap \ell_k$, and H is the set of humans of π (π itself if a human, otherwise its holders).

The checker

$\mathrm{check}(\mathrm{EDB}, c)$ accepts iff each of the following holds by lookup in the EDB snapshot of digest d: (1) the charter and every chain edge are facts; (2) the chain is contiguous from a_0 to g and revisits no agent; (3) every w_i is a valid InScope path for s; (4) ℓ^* is the meet of the path levels; (5) $\mathrm{Req}(s)$ is exactly the set of requirements of s and $\ell^* \geq$ every level in it; (6) $(x, \ell_x) \in \mathrm{Req}(s)$ with $\ell_x \geq \ell$; (7) H is exactly the set of humans of π and $H \neq \emptyset$; (8) $\mathrm{digest}(c)$ is correct; (9) d equals the digest of the EDB, or of a trusted snapshot supplied by the verifier. The checker runs no fixpoint and reads no derived predicate. Reference implementation: `charterion_cert.py` (Chapter 22).

Condition 11.7 (**CC7**: Certified envelope)**.** When the certificate profile is in use, every approved envelope entry $e = \langle g, x, \ell, \mathit{appr}\rangle \in \mathcal{E}_t$ SHALL carry an authority certificate that the checker accepts for $\langle g, x, \ell'\rangle$ with $\ell' \geq \ell$ against the current model snapshot:

$$\textsc{Uncertified}(e) \leftarrow e \in \mathcal{E}_t,\ \mathit{status}(e) = \mathit{approved},\ \neg\exists c\ \mathit{cert}(e, c).$$
$$\textsc{InvalidCertificate}(e) \leftarrow e \in \mathcal{E}_t,\ \mathit{status}(e) = \mathit{approved},\ \mathit{cert}(e, c),\ \neg\mathrm{check}(\mathrm{EDB}_t, c, \langle g, x, \ell\rangle).$$

Decision-log records (**Inv-5**, **CC6**) SHOULD carry the certificate of the entry that produced their outcome (Appendix L).

Proposition 11.2 (Soundness of the checker)**.** *If* $\mathrm{check}(\mathrm{EDB}, c)$ *accepts for the claim* $\langle g, x, \ell\rangle$*, then in the least model of the model whose EDB has digest* d*:* $\textsc{Auth}(g, s, \ell^*)$*,* $\textsc{Covers}(g, s)$*,* $\textsc{Need}_{\geq}(g, x, \ell)$ *and* $\mathrm{Acc}(g, h)$ *for every* $h \in H$*; and if the chain is empty,* g*'s accountability derives from its own charter.*

Proof. Each accepted check is one rule application of Figure 7.2. The charter fact with w_0 gives $\textsc{Auth}(a_0, s, \ell_0)$ by (A1) and (S1)–(S3); each chain edge with its witness gives $\textsc{Auth}(a_i, s, \ell_0 \sqcap \ldots \sqcap \ell_i)$ by (A2); hence $\textsc{Auth}(g, s, \ell^*)$. Check (5) gives $\neg\textsc{Shortfall}(g, s)$, hence $\textsc{Covers}(g, s)$ by (C1)–(C2); check (6) gives $\textsc{Need}(g, x, \ell_x)$ by (N1) and $\textsc{Need}_{\geq}(g, x, \ell)$ by (N2). Check (7) gives $\mathrm{Acc}(a_0, h)$ for $h \in H$ by (K1)–(K2), and (K3) carries it along the chain to g. □

Proposition 11.3 (Completeness)**.** *If* $\textsc{Need}_{\geq}(g, x, \ell)$ *holds in the least model of* M *and* $\neg\textsc{Orphan}(g)$*, then a certificate exists that* check *accepts against* M*; the reference builder constructs one by breadth-first search over the delegations whose scope contains* s*, from the charters whose scope contains* s*.*

Proof. $\textsc{Need}_{\geq}(g, x, \ell)$ gives by (N1)–(N2) a step s with $\textsc{Covers}(g, s)$ and $\textsc{Requires}(s, x, \ell_x \geq \ell)$; Covers gives $\textsc{Auth}(g, s, \ell^*)$ with $\ell^* \geq$ every requirement of s. Auth derives by one (A1) application followed by finitely many (A2) applications, i.e. by a charter whose scope contains s and a delegation path to g whose scopes contain s and whose meet is $\geq \ell^*$; the search finds a shortest such path, which is simple because the meet along a cycle never increases. $\neg\textsc{Orphan}(g)$ gives a human in $\mathrm{Acc}(g, \cdot)$, derived along some such path from a charter whose principal has humans; the builder prefers those charters, and if none existed g would be an Orphan. □

Corollary 11.1 (Theorem 11.1 auditable per decision)**.** *Under* ***CC1****,* ***CC5*** *and* ***CC7****, for every permit or escalate record in* $\mathcal{L}$ *whose envelope entry carries certificate* c*, the charter, delegation chain, step, need and accountable humans whose existence Theorem 11.1 asserts are the components of* c*, and a verifier confirms them in* $O(|c|)$ *lookups against the snapshot of digest* d *without running the reasoner.*

Proposition 11.4 (Complexity)**.** check *runs in* $O(k + |\mathrm{Req}(s)| + |H|)$ *lookups for a chain of length* k*. Building one certificate runs in* $O(|\textsc{Charter}_s| + |\textsc{Delegation}_s| \cdot |A|)$*, the subscript restricting to scopes containing* s*; certifying the whole charter-derived envelope runs in* $O(|\textsc{Need}| \cdot (|\textsc{Charter}_s| + |\textsc{Delegation}_s| \cdot |A|))$*.* ***CC7*** *on a register* $\mathcal{E}_t$ *runs in* $O(\sum_e |c_e|)$*.*

Remark (What certificates do not do). A certificate proves that an entitlement *follows from* the recorded charters; it does not prove that the recorded charters are the organisation's will (that is gate G2 and **CC5**), nor that the snapshot digest is authentic (that requires the digest to be signed by the provisioning pipeline of Chapter 22, which is a binding concern, Appendix K). Certificates bind to a snapshot: any change to the model invalidates every certificate, by design; re-issue is part of phase D5. **Inv-3** and **X-Inv-1** are preserved: a certificate is derived from charters approved by humans, its accountable set H must be non-empty, it cannot be issued by the agent it concerns, and the Authority Request Protocol does not consume certificates (Proposition 16.1 unaffected; Corollary 8.2).

Table 11.2. Cross-plane coherence condition added in v1.1 (continuation of Table 11.1)

Id	Name	Data compared	Defect	Axiom
CC7	Certified envelope	Envelope entries and decision-log records vs. the derivation from the model snapshot, checked without the reasoner	Uncertified, InvalidCertificate	X1 no authority without charter (evidence side); X6 no answer without warrant

Part III

The Charterion Continuous Method

12 Overview of the method

12.1 From a cycle to a loop

The TOGAF Architecture Development Method is a cycle of phases traversed by a project: a vision, three domain architectures, opportunities and solutions, migration planning, implementation governance, change management. Its unit of work is the architecture engagement, and its tempo is that of transformation programmes. An agentic-driven enterprise changes at a different tempo: charters are added when an agent is onboarded, delegations appear when an orchestrator learns a new tool, grants drift when a platform team responds to a ticket, and the substrate is re-observed every time an agent acts. The Charterion Continuous Method therefore replaces the cycle by *loops that run at the tempo of the thing they govern*: a design loop that runs on every change to the model, an operation loop that runs on every action, and a boundary loop that runs on every relationship—joined by two bridges that carry authority forward and evidence back (Figure 12.1).

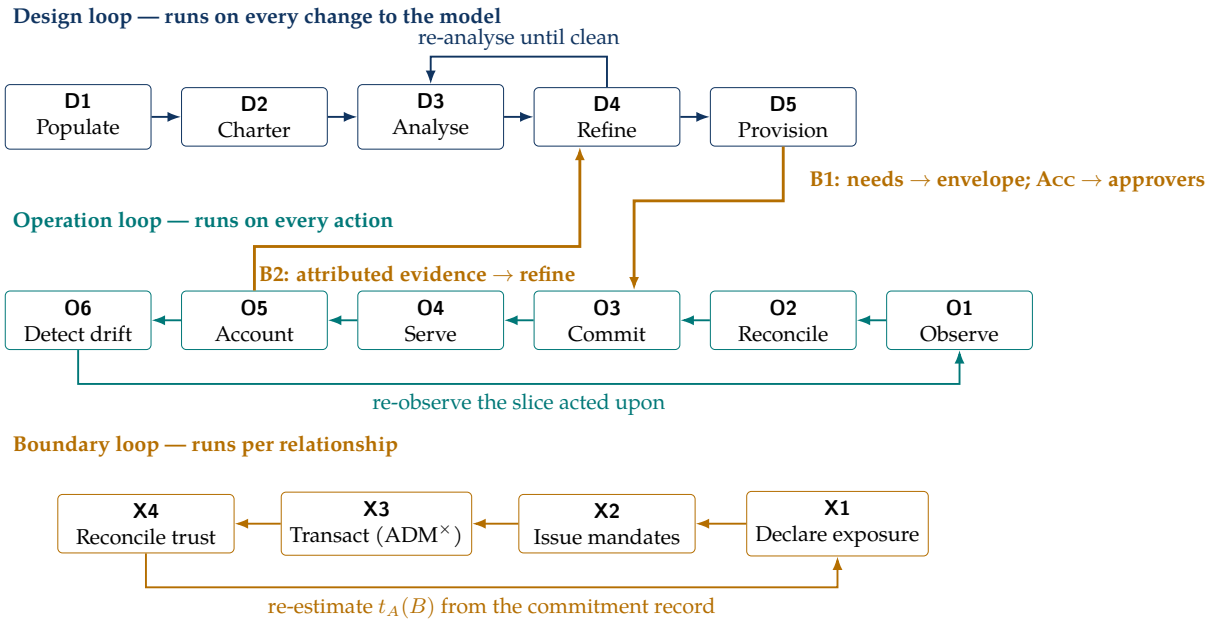

Figure 12.1. The Charterion Continuous Method: three loops and two bridges. Design phases D1–D5 are the five steps of El Hassani (2026e); operation phases O1–O6 are the Substrate Lifecycle of El Hassani (2026c); boundary phases X1–X4 follow El Hassani (2026d); the bridges are new in v1.0. The operation and boundary loops are drawn right to left so that the bridges do not cross. Not drawn: D5 also emits candidate mandates consumed by X2 (**CC3**, **CC5**), and O5 records the commitments from which X4 re-estimates trust.

12.2 Structure of a phase description

Each phase in Chapters 13–15 is described by the same template: *purpose*; *inputs* (artefacts consumed); *activities*; *outputs* (artefacts produced or updated); *roles* (from Chapter 18); *automated*

checks (the conditions a tool evaluates on exit); and *exit criteria*. The artefacts are those of Chapter 21; the checks are the conditions of Part II.

12.3 Slice discipline

Slice discipline

The method SHALL be applied to a declared *slice* of the organisation: a set of capabilities or processes, the resources they touch and the agents chartered on them. A slice has one design-plane conformance level, one execution-plane conformance level and, per relationship, one exposure level (Chapter 25). The level of a slice is the level of its least conformant layer. Slices MAY overlap on resources; where they do, the contention condition **WF5** is evaluated on their union.

The slice is the unit of adoption, of conformance and of reporting. It replaces the enterprise-wide target architecture of classical methods with something an organisation can actually bring to a stated level: a payments slice at execution Level 3 and design level D3, a customer-service slice at Level 1 and D1. The Substrate Framework's evaluation showed that the value of the substrate rises steeply with the level in the slice being acted upon and not at all with coverage elsewhere (El Hassani, 2026c); the slice discipline follows.

13 The design loop

The design loop is the five-step method of the fifth article, given here as phases with the v1.0 additions. It runs on every change to the design-plane model: a new agent, a changed charter, a reorganisation that vacates a role, a new tool that adds a resource. In a repository with a continuous-integration pipeline the loop is a check that gates every commit (Chapter 22).

13.1 D1 — Populate the core

Purpose. Obtain the business, application and technology facts the analysis needs. *Inputs.* The ArchiMate repository or its equivalent; process models; the identity-management export of agent entitlements. *Activities.* Project the repository onto the core predicates (Section 6.4); mark Agentic steps from the business's automation decisions and Reserved steps from its human-reservation decisions; import provisioned Grant facts from identity management; record the source and date of each import. *Outputs.* The core model; the Principal & Vacancy Register. *Roles.* Agency architect (owner), substrate owner, domain authorities. *Automated checks.* Referential integrity (every step has a process, every requirement names a resource, every role holder is a human); no step both agentic and reserved; import freshness within the declared window. *Exit criteria.* Every agentic step has at least one Requires fact; every role named by a charter is in the register with its current holders or marked vacant.

Rationale. Requires is the fact that carries the analysis: a step without requirements is covered by any charter and needs nothing, so the derived envelope of an agent on such steps is empty and **WF3** cannot judge its grants. Process models rarely state which resources a step touches and at which level; D1 is where that gap is closed, typically by the domain authority of each process. The projection study of the fifth article measured what is lost when it is not closed (El Hassani, 2026e).

13.2 D2 — Charter the agents

Purpose. State, as facts, what each agent may do and who answers for it. *Inputs.* The core model; the agent platform's inventory; the business's intent for each agent. *Activities.* For each agent create Agent; for each intended authority create Charter with scope, level and principal, following the granularity rule (Section 7.5); record Delegation for each agent-to-agent hand-off the design intends; declare guards—SoD pairs from the business's control framework, Precedence for known shared resources, Oversight for every intended autonomous agent, Escalation for every write-level agent; attach Validity, Review and Cap where the temporal and quantitative profiles are in use; write the Charter Rationale Record for every write-level charter. *Outputs.* Agent Register; Charter Register; Delegation Register; Guard Register; Charter Rationale Records. *Roles.* Agency architect, principals (who accept accountability), overseers, risk and security. *Automated checks.* Every charter names an existing agent, scope and principal; every write-level charter has a rationale record; profile completeness (**WF10**-Unbounded). *Exit criteria.* Every agent in the platform inventory has at least one charter or is recorded as decommissioned.

Agent onboarding procedure

A new agent SHALL enter the organisation through D2: (1) the sponsoring business owner states the steps it is to perform and the level at each; (2) the agency architect drafts charters at the narrowest scope covering those steps, and delegations only where the agent hands work to existing agents; (3) the principal accepts accountability in writing (the Charter Rationale Record); (4) D3 is run and every defect resolved or accepted; (5) D5 provisions the derived needs and nothing else; (6) the agent's platform identity is bound to its AGENT constant. No agent SHALL receive an entitlement before step 5.

13.3 D3 — Analyse

Purpose. Compute the least model and evaluate every condition. *Inputs.* The complete design-plane model; the envelope, mandate and decision registers for the coherence conditions. *Activities.* Run the reasoner (Chapter 22) at the instant of analysis; produce the Well-formedness Report (**WF1–WF12** with witnesses), the Coherence Report (**CC1–CC6**) and the Agency Debt Dashboard (Chapter 20); compare with the previous run and list new, resolved and persisting defects. *Outputs.* Well-formedness Report; Coherence Report; Agency Debt Dashboard; derived Authority Matrix, Need-vs-Grant Matrix and Contention Matrix. *Roles.* Agency architect; tool. *Automated checks.* All conditions of Part II. *Exit criteria.* None—D3 always exits to D4.

13.4 D4 — Refine

Purpose. Resolve or accept every defect. *Inputs.* The reports of D3; the evidence records arriving over bridge B2. *Activities.* For each defect, choose one of the resolutions of Table 13.1 or record an accepted defect with rationale, owner and review date in the Accepted-Defect Register; re-run D3 until the report contains only accepted defects; apply evidence-gated level changes (Section 18.8) proposed by B2. *Outputs.* Updated registers; Accepted-Defect Register; charter lifecycle transitions. *Roles.* Agency architect proposes; principals and the Agency Governance Board decide (Chapter 18). *Automated checks.* D3 re-run; every accepted defect has an owner and a review date not in the past. *Exit criteria.* The Well-formedness and Coherence Reports contain no unaccepted defect.

13.5 D5 — Provision from the model

Purpose. Make the architecture the source of entitlements and of the run-time envelope. *Inputs.* The least model of the refined design plane. *Activities.* Emit the derived NEED set as the target entitlement set for identity management and compare with the provisioned grants (the Grant Reconciliation Ledger); construct the charter-derived envelope (Definition 9.5) and submit it to the approvers named by Acc for commitment in O3 (bridge B1); where the boundary plane is in use, derive candidate mandates for X2. *Outputs.* Grant Reconciliation Ledger; proposed envelope entries with provenance; candidate mandates. *Roles.* Agency architect emits; identity-management and agent-platform teams apply; approvers commit in O3. *Automated checks.* After provisioning, **WF3** and **CC1–CC2** hold. *Exit criteria.* The Grant Reconciliation Ledger shows no unaccepted difference between needs and grants.

Rationale. D5 inverts the direction of current practice, in which entitlements are provisioned first and the architecture, if it exists, is documentation. The provisioning target is computed, not administered, and **WF3** is the continuous check that it stays so. The fifth article's evaluation found that an ordinary first-pass model contains grants added by hand that no covered step requires and entitlements that a chartered agent lacks; D5 removes both, and does so on every run.

Table 13.1. Standard resolutions of defects in D4

Defect	Standard resolutions	Notes
WF1	Assign a holder to the vacant role; re-principal the charter to a current human or held role; retire the agent	Never resolve by deleting the delegation chain: the agent's authority remains
WF2	Add a charter covering the step; widen an existing charter's level to meet the step's requirement; un-designate the step	Check for a new **WF4**, **WF5** or **WF8** before accepting a wider charter
WF3	Shadow: remove the grant or add a charter that justifies it. Missing: provision the grant (D5)	Shadow grants added "by hand" are the commonest finding on first analysis
WF4	Split the covering charter so that two agents cover the pair; re-scope to a step	Two agents under the same principal still satisfy **WF4**; **WF9** and the Agency Governance Board judge whether that is acceptable
WF5	Declare a Precedence; re-scope one charter so the steps are no longer concurrently covered; lower one agent to *propose* on the resource	Precedence is a coordination decision of the business, not of the platform team
WF6	Lower the delegation level to the delegator's authority; add the missing authority to the delegator (rare)	Over-delegation usually indicates an orchestrator designed before its own charter
WF7	Add Oversight; lower the charter to *commit*	Adding oversight raises the overseer's span (**WF9**)
WF8	Narrow or split the charter so the reserved step is out of scope; lower the charter to *propose* on that scope	Repair at the charter, never at the delegation (Proposition 8.1)
WF9	Name an Escalation human; fill the vacancy; redistribute oversight to lower a span; raise β with justification	Raising β is a governance decision recorded in the Conformance Statement
WF10	Add validity and review dates; re-approve the charter (which resets the review date); retire the expired charter	Re-approval is a lifecycle gate (Section 18.5)
CC1	Remove the envelope entry; add a charter that grounds it	The former is the default; the latter needs a principal
CC2	Approve the missing entry (D5/B1)	Usually a provisioning backlog
CC3	Narrow the mandate; raise the cap of the grounding charter (governance decision)	—
CC4	Raise the slice's execution-plane level before conferring write authority; lower the exposure level	Sequencing decision of the roadmap (Chapter 25)
CC5	Re-approve the entry by an accountable human; update Acc by fixing the charter's principal	—
CC6	Add the missing charter (modelling defect) or open an incident (agent acting outside its architecture)	Triage by the substrate owner

Table 13.2. Standard resolutions of the v1.1 defects (continuation of Table 13.1)

Defect	Resolutions, in order of preference	Acceptance code (if any)
ConfigDrift(a, m)	Re-review: record Reviewed(m, κ) for the running κ after evaluating the change (G2 re-entry); roll the agent back to the reviewed configuration	`config-change-minor` (time-boxed)
Unreviewed(m)	Record the reviewed digest at approval	`pre-profile`
Uncharted(i)	Stop the instance (O6 incident); or restore or charter the type through the protocol of Chapter 16	none
OverInstantiated(a)	Scale down; or raise Cardinality by Charter Review decision	none
OverloadedI(h)	Add overseers; reduce instances	as **WF9**
UnstaffedReserved(s)	Assign a staffed role (Performs); fill the vacancy; re-designate the step (business decision)	none
CapAmplification(a, u)	Reduce delegated caps; split the delegator's cap explicitly; raise the delegator's cap (G2)	`sequential-delegatees`
UncappedDelegation(a, b, σ)	Declare DelegationCap	none
UnmetObligation(o, a, s)	Lower the charter to *commit*; reserve the step (**WF8**) and staff it (**WF15**); or correct the obligation's scope with the regulatory owner	none
OutOfPerimeter(a, b)	Declare the edge (architecture decision, G2); remove the delegation	`runtime-discovered` (time-boxed)
DelegationCycle(a), DepthExceeded(a, b)	Break the cycle; shorten the chain	none
BlastExceeded(a)	Re-scope write-level charters to steps; split the agent; add compensation (**WF12**); raise the bound by review with justification	`bound-review`
Uncertified(e), InvalidCertificate(e)	Re-run D5 (certificates are emitted with the envelope); if the derivation fails, the entry is not grounded—retire it or repair the model (**CC1** also reports it)	none

14 The operation loop

The operation loop is the Substrate Lifecycle (Section 9.4) as it is run by the substrate owner. It runs on every action and every observation, and its phases are largely automated; the human work is concentrated in O3 (approval) and in the decisions that O4 escalates.

14.1 O1 — Observe

Systems, agents and people emit observed assertions with source, confidence and validity into the anchor, affordance and coordination layers. Agents SHALL re-observe the slice they acted upon after every permitted action (currency in the acted-upon slice, **SR7**). Observed envelope content MAY be stored as a proposal but never authorises (**Inv-2**). *Outputs:* the Assertion Store. *Checks:* every assertion has a finite validity (**Inv-4**); no envelope assertion has an agent as source (**Inv-3**).

14.2 O2 — Reconcile

Names are resolved against the anchor layer; conflicting observations from federated sources are resolved by the domain's precedence rule or kept as two assertions with reduced confidence; derived assertions are recomputed with the minimum confidence and intersected validity of their premises; the design plane's core facts are compared with the anchor layer and differences are reported to D1. *Outputs:* reconciled Assertion Store; discrepancy list for D1. *Checks:* no identifier resolves to two elements above θ_{id}.

14.3 O3 — Commit

The humans named by Acc approve the envelope entries proposed by D5 (bridge B1) and the ownership assertions of the coordination layer; approval records the approver and produces $\varepsilon = \mathit{approved}$. Nothing else requires approval. A proposed entry whose approver is not in $\mathrm{Acc}(g, \cdot)$ SHALL NOT be committed (**CC5**). *Outputs:* approved envelope; Decision Log entry for each approval. *Checks:* **Inv-1**, **Inv-3**, **CC1**, **CC2**, **CC5**.

14.4 O4 — Serve

The admissibility function answers requests (Algorithm 1). *Escalate* outcomes are routed to $\mathrm{Esc}(g, \cdot)$; *unwarranted* outcomes are routed to the domain authority for re-observation or a human answer; *deny* outcomes are returned with their supporting assertions so that the agent can explain the refusal. *Outputs:* decisions; escalations. *Checks:* response within the slice's declared latency; the parameters $\theta_{\mathrm{id}}, \theta_{\mathrm{act}}$ are those of the Conformance Statement.

14.5 O5 — Account

Every decision is appended to the Decision Log with its supporting assertions and minimum warrant (**Inv-5**); every state-changing action is recorded as a commitment in the coordination

layer (and, across a boundary, in both substrates); reversals are recorded against the original decision. Decisions are attributed to charters (**CC6**) and accumulated into the Evidence Record of each charter: counts of permit, deny, escalate and unwarranted outcomes, escalations accepted and refused by the human, and reversals. *Outputs:* Decision Log; Commitment Ledger; Evidence Records. *Checks:* **CC6**; the log is append-only.

14.6 O6 — Detect drift

Outcomes are compared with what the estate later shows: an action permitted on a precondition later found false, an affordance observed but not invocable, an identity that resolved to the wrong element. Validity intervals and confidences of the responsible sources are re-estimated; systematic drift on a charter's scope is reported to D4 over bridge B2. *Outputs:* re-estimated warrant; drift report. *Checks:* the drift-detection cadence of the Conformance Statement is met.

14.7 The two bridges

New in Charterion v1.0

Bridge B1 (D5 → O3). The derived needs of the refined design plane become the proposed envelope; the accountable humans become the approver set; the charter identifiers become the provenance of every envelope entry. B1 is the mechanism by which the design plane is the *source* of run-time authority; **CC1**, **CC2** and **CC5** are the conditions that it has been crossed correctly.

Bridge B2 (O5/O6 → D4). The attributed Evidence Records and drift reports return to the refine phase, where they drive three kinds of change: *narrowing* (a charter whose scope produces many denials or reversals is too broad), *widening* (an agent whose needs produce many unwarranted or unserved outcomes is under-chartered or the slice under-observed), and *level changes* under the evidence-gated promotion rule of Section 18.8. B2 is the mechanism by which the architecture learns; **CC6** is the condition that no evidence is lost on the way.

15 The boundary loop

15.1 X1 — Declare exposure

The receiving organisation declares, per relationship, its exposure level (Section 10.3) and publishes its external affordance inventory and boundary envelope as approved content. **CC4** is checked: the exposure level of a relationship SHALL NOT exceed what the execution-plane level of the exposed slice supports. *Outputs:* Exposure Declaration. *Roles:* approvers of the exposing organisation; Boundary Review.

15.2 X2 — Issue mandates

The issuing organisation issues mandates to its agents from the candidates derived in D5: operations within the agent's needs (**CC3**), bands within the charter's caps, validity within the charter's, issuer in $\mathrm{Acc}(g, \cdot)$ (**CC5**), counterparties named. The Mandate Register records the grounding charters. *Outputs:* Mandate Register; signed mandates. *Roles:* the accountable humans (issuers); Boundary Review.

15.3 X3 — Transact

Requests across the boundary are decided by $\mathrm{ADM}^{\times}$ (Definition 10.5); escalations are decided on the side that raised them; every state-changing action creates a commitment with a shared referent in both substrates. *Outputs:* decisions in both Decision Logs; Commitment Ledger entries. *Checks:* **X-Inv-1**, **X-Inv-2**; both sides' logs carry the shared referent.

15.4 X4 — Reconcile trust

On the declared cadence each organisation re-estimates $t_A(B)$ from its own commitment record (satisfied, violated, cancelled) under **X-Inv-2**; changes propagate to κ_{eff} and hence to admissibility (X-P3). Systematic violations by a counterparty's agents are reported to the counterparty as evidence for *its* D4. *Outputs:* Trust Weight Register; counterparty evidence report. *Roles:* approvers; Boundary Review.

16 The Authority Request Protocol

New in Charterion v1.0

Everything in the method so far flows from people to agents: humans charter, the analysis checks, the substrate serves. Agents in an agentic-driven enterprise, however, discover needs that no charter anticipated—a new tool, a resource that a process change now requires, a counterparty that did not exist when the mandate was issued. Today such needs are met outside the architecture: a platform team adds a grant, an orchestrator is given a broader scope. The Authority Request Protocol (ARP) gives the agent a way to *ask the architecture* for authority, and gives the architecture a way to answer through its human gates, so that the model remains the single origin of authority even when the initiative comes from the agent. It is the constructive counterpart of **Inv-3** (non-self-authorisation): an agent may propose any assertion about its own authority; it may approve none.

16.1 Requests

Definition 16.1 (Authority request)**.** An *authority request* is a tuple $\rho = \langle g, \sigma, \ell, J, W, t \rangle$ submitted by agent g: $\sigma \in \Sigma$ the scope requested, $\ell \in \mathbb{L}$ the level requested, J a justification consisting of the request q that was refused (or the step s the agent was asked to perform and cannot cover), the admissibility outcome and its supporting assertions W from the decision log, and t the instant of submission. A request is well formed if g is a chartered agent, σ exists in the core model, and J references a record of $\mathcal{L}$ with outcome $deny$ or $unwarranted$ for g, or an uncovered agentic step assigned to g's scope.

A request is not a charter and confers nothing. It enters the design loop as a *Proposed* charter $\textsc{Charter}(m_\rho, g, \sigma, \ell, \pi)$ whose principal π is, by default, the principal of g's existing charter on the nearest enclosing scope, or its overseer where none exists; the request is recorded in the Charter Register with provenance σ = *agent-requested* and the justification J as its Charter Rationale Record.

16.2 Protocol

1. **Refusal or shortfall.** The substrate returns *deny* or *unwarranted* to g, or the design loop reports $\textsc{Uncovered}(s)$ for a step in g's scope.
2. **Submission.** g submits ρ. The request is rate-limited per agent and per charter (a parameter of the Conformance Statement); repeated requests for the same (σ, ℓ) within the window are merged.
3. **Pre-analysis (automatic).** D3 is run on $M \cup \{\textsc{Charter}(m_\rho, \ldots)\}$. The Well-formedness and Coherence Reports are computed *as if* the charter were approved, and the delta in every **WF**, **CC** and agency-debt component is attached to the request. A request whose pre-analysis shows any of **WF1**, **WF4**, **WF8**, **WF11** or **WF12** failing is *returned* to the agent with the defect as reason; no human is consulted.
4. **Decision (human).** A request that passes pre-analysis is presented to the proposed principal with the delta and the justification. The principal may approve as proposed, approve at a lower level or narrower scope (the request then becomes the corresponding Proposed charter and re-enters at step 3), or refuse with a reason. A request at *autonomous* SHALL additionally pass the Agency Governance Board. The decision is recorded.

5. **Provisioning.** An approved request proceeds through gates G2–G4 of the charter lifecycle like any charter; the envelope is re-derived (bridge B1) and the agent's next request is admissible.
6. **Learning.** Returned and refused requests are counted per charter in the Evidence Record; a charter that generates repeated well-founded requests is a candidate for re-scoping at D4, and a charter that generates repeated ill-founded ones is a candidate for degradation.

16.3 Properties

Proposition 16.1 (Non-self-authorisation is preserved)**.** *Under the protocol, for every instant t the set of* Auth *tuples in the least model of M_t is derived only from charters and delegations whose lifecycle state is* Active*, and no transition into* Approved *is performed by an agent. Hence an agent's request changes its authority only through a decision recorded to a human, and* ***Inv-3*** *holds at every instant.*

Proof. A request creates a charter in state *Proposed*. By the lifecycle (Section 18.5) only *Active* charters contribute to M_t; the transition Proposed $\rightarrow$ Analysed is automatic but confers nothing, and Analysed $\rightarrow$ Approved is gate G2, whose decision right is the principal's (Table 18.2). No rule of the protocol assigns G2 to an agent. The delegation case is identical, since a requested delegation is a requested charter on the delegatee bounded by the delegator's authority (**WF6**). □

Proposition 16.2 (Pre-analysis is sound)**.** *If the pre-analysis of ρ reports no defect, then approving m_ρ as proposed yields a model whose defect sets are exactly those reported, because the pre-analysis is the analysis of $M \cup \{m_\rho\}$ and the analysis is deterministic (Proposition 7.1).*

Rationale. Access-request workflows in identity governance, step-up authorisation, and permission requests in electronic institutions for multi-agent systems are the protocol's cousins; the intent-declaration-before-grant sequence of the 2026 identity-governance literature (Chapter 24) is its nearest contemporary. None of them computes what the architecture would look like if the request were granted, and none binds the decision to the enterprise model's conditions and lifecycle; that pre-analysed, model-bound form is what the protocol adds. It changes what an agent can do when it is refused: instead of failing silently, retrying, or being given a broader grant by a helpful engineer, it produces a structured, pre-analysed proposal that a named human decides on the basis of what the architecture would look like if the proposal were accepted. Three consequences are worth stating. The shadow-grant path is closed, because the only way to obtain an entitlement is a charter and the only way to obtain a charter is a decision. The design loop acquires a second source of input besides the architect, without losing its gate. And the organisation obtains a measure it did not have: the rate at which its agents ask for authority they were not given is, per slice, a direct reading of how well the charters fit the work. The protocol is what makes the substrate an *interlocutor* of the agent rather than only its gate; it is, in the author's view, the construct by which an enterprise model stops being documentation and becomes an institution.

16.4 Artefacts and metrics

The protocol adds one register, the *Authority Request Register* (request, agent, scope, level, justification, pre-analysis delta, decision, decider, timestamps), and four run-time metrics per charter and slice: requests submitted, returned by pre-analysis, refused, approved (as proposed / narrowed). The reference implementation exposes the pre-analysis as a function on a model and a candidate charter (Chapter 22); the routing of the human decision is the organisation's workflow system's.

New in Charterion v1.1

Positioning of the protocol (informative). Run-time step-up through a human exists as a mechanism in three works retrieved for v1.1: the OWASP Agent Control Standard's `ask` outcome, routed to a human approver, and its `intent_extension`; the CIBA-based flow of the IETF AIMS draft (§10.7); and the escalation to user confirmation of South et al. (2025). Each grants, for a session, a right the agent asked for. The Authority Request Protocol differs in four respects, and claims only these: the request is pre-analysed against the *design model* (**WF1**, **WF4**, **WF8**, **WF11**, **WF12**, and in v1.1 **WF17** and **WF18**) before a human sees it, so that what the model can refuse is never decided by a person and what the model cannot refuse is never decided by the model; it is decided through the charter lifecycle of Chapter 18; what results is a *charter*, whose entitlements are derived and certified (Section 11.5), not a session grant; and it is registered. A binding may implement step 1 of the protocol over the Agent Control Standard's `ask` channel (Appendix K).

17 Coexistence with the TOGAF ADM

Charterion does not require an organisation to abandon the ADM. The design loop maps onto the ADM as follows, and the full mapping is tabulated in Appendix F. D1 is performed within Phases B, C and D (business, information systems and technology architectures), whose deliverables are the source of the core model, and the AGENTIC/RESERVED designations are decisions recorded in Phase B. D2 is a new activity of Phase B (charters and guards are business-architecture content) with inputs from the Preliminary Phase (principles, including the six axioms) and from Requirements Management. D3 and D4 are the compliance-assessment activity of Phase G (Implementation Governance) made continuous and automatic. D5 corresponds to the delivery of Phase G and the transition into operations. The operation loop has no ADM counterpart—the ADM ends at Phase H (Architecture Change Management), which Charterion replaces for the agentic slice by the continuous loops and bridge B2. The boundary loop corresponds to no ADM phase and extends the Enterprise Continuum outward to the counterparties.

Two ADM habits must change. First, the "target architecture" of a slice is replaced by a *target conformance level* with the roadmap of Chapter 25; there is no future-state model of the agents, because their population changes too fast, only conditions the population must satisfy at every instant. Second, the architecture board's review of solutions is replaced, for the agentic slice, by the Agency Governance Board's gates on the charter lifecycle (Chapter 18), which are exercised by tools on every change and by people at the transitions that need judgement.

Part IV

Governance

18 The governance model

18.1 Governance as computation

Classical EA governance is exercised by a board that reviews documents at milestones. Charterion governance is exercised by *conditions on a model* that a tool evaluates on every change, and by *people* at the transitions of a lifecycle that require judgement: accepting accountability, approving an envelope, accepting a defect, promoting a level. The design principle is that every control is either a condition of Part II—in which case it is checked automatically and its failure is a reported defect—or a decision right assigned to a named role at a named lifecycle gate—in which case it is exercised by a person and recorded. Nothing in Charterion governance is a prose control that depends on someone remembering to apply it.

18.2 Roles

A charter enters the lifecycle either from the agency architect (D2) or from an agent through the Authority Request Protocol (Chapter 16); the gates are the same in both cases, and the decision rights below apply regardless of who proposed.

Table 18.1 defines the roles. They are *roles*, not positions: one person may hold several, and the substrate owner is in most organisations the EA function itself, which the survey found to be the function practitioners expect to own the substrate (El Hassani, 2026b).

Table 18.1. Roles of the Charterion governance model

Role	Responsibility	Where it appears in the model
Agency architect	Owns the design plane: populates the core, drafts charters and guards, runs the analysis, proposes refinements, emits provisioning targets	Author of Charter, Delegation, guard facts; runs D1–D5
Substrate owner	Owns the execution plane: the assertion store, the admissibility service, the decision log, the lifecycle cadence, the parameters $\theta_{\mathrm{id}}, \theta_{\mathrm{act}}$	$\mathcal{S}$; O1–O6; Conformance Statement
Domain authority	Answers for the anchor and affordance content of a domain; resolves conflicts and unwarranted outcomes in it	Source σ of observed assertions; per-domain precedence rule
Principal	The human, or holder of the role, named by a charter; accountable for what the agent does under it	π in Charter; Acc
Approver	Commits envelope and ownership assertions; issues mandates; must be in $\mathrm{Acc}(g, \cdot)$ for agent g	$\mathcal{H}$; h_μ; **CC5**
Overseer	Monitors and can halt an agent; decides its escalations	Oversight, Escalation, Esc
Agent platform team	Runs the agents; applies provisioning targets to identity management; emits observations and decision requests	Grant; O1; O4 clients
Auditor	Reads the decision log and the registers; replays decisions; verifies the Conformance Statement	**Inv-5**; **CC6**
Agent	A non-human principal; may observe, request and propose; may not approve	$\mathcal{G}$; Agent

18.3 Bodies

Agency Governance Board
: extends the architecture board with risk, security, legal and business owners. It decides: acceptance of defects with enterprise-level consequence (any **WF1**, **WF4**, **WF8** or **CC5** acceptance); write-level capability charters (granularity rule); the span bound β; the thresholds of evidence-gated promotion; changes to the Conformance Statement. It meets on a cadence and *on demand when a gate requires it*; most transitions never reach it.

Charter Review
: is the standing forum of agency architect, principals and overseers that exercises the routine gates of the charter lifecycle (Section 18.5) and reviews the Agency Debt Dashboard.

Boundary Review
: is the forum of approvers and the substrate owner that declares exposures, approves mandates and re-estimates trust weights; it includes legal and commercial owners of the relationships.

18.4 Decision rights

Table 18.2 assigns decision rights. *D* decides, *P* proposes, *C* is consulted, *I* is informed, *T* is performed by the tool. A decision that a tool performs is one whose outcome is determined by the model; the human roles are informed of the result and may not override it except by changing the model.

Table 18.2. Decision-rights matrix

Decision	Arch.	Subst.	Princ.	Appr.	Overs.	Board	Audit	Tool
Designate a step agentic or reserved	P	I	D	–	C	I	–	–
Create or change a charter (read/propose)	D	I	C	–	–	I	–	T
Create or change a charter (commit/autonomous)	P	I	D	–	C	I	I	T
Write-level charter on a capability	P	I	C	–	C	D	I	T
Accept accountability as principal	–	–	D	–	–	I	–	–
Declare SoD, precedence	P	C	D	–	–	I	–	T
Name oversight / escalation human	P	–	C	–	D	I	–	T
Evaluate **WF1–WF12**, **CC1–CC6**	I	I	I	I	I	I	I	D
Accept a defect (routine)	P	C	D	–	C	I	I	–
Accept a defect (**WF1**, **WF4**, **WF8**, **CC5**)	P	C	C	–	C	D	I	–
Provision derived needs	P	–	I	–	–	–	I	T
Approve an envelope entry	P	C	–	D	–	–	I	–
Decide an escalation	–	–	I	–	D	–	I	–
Promote a charter level	P	C	C	–	C	D	I	T (gate)
Demote or suspend a charter	D	C	I	–	D	I	I	T (stale)
Set β, θ_{id}, θ_{act}, promotion thresholds	P	P	–	–	–	D	I	–
Declare an exposure level	C	P	–	D	–	C	I	–
Issue a mandate	P	–	–	D	–	I	I	T (**CC3**)
Re-estimate a trust weight	–	P	–	D	–	I	I	T

18.5 The charter lifecycle

New in Charterion v1.0

The charter lifecycle is new in v1.0. It gives every charter a state, and every state transition a gate consisting of conditions (checked by the tool) and decisions (taken by a role). Provisioning follows the state: only *Active* charters contribute to the envelope in force.

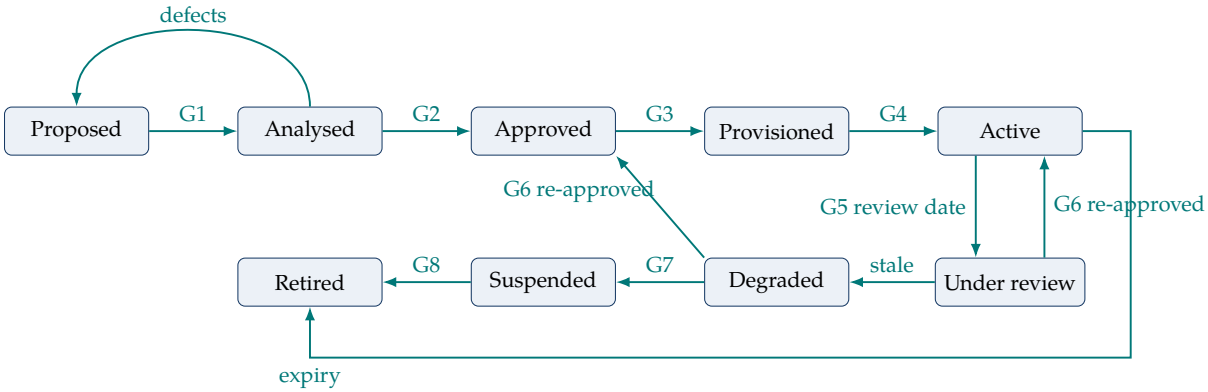

Figure 18.1. Charter lifecycle states and gates. Gate G7 (suspension) also applies directly from *Active*; the edge is omitted for legibility.

Table 18.3. Gates of the charter lifecycle

Gate	Transition	Conditions (tool)	Decisions (role)
G1	Proposed → Analysed	Referential integrity; rationale record present for write levels; profile completeness	Agency architect submits
G2	Analysed → Approved	No unaccepted **WF1**–**WF12** defect involving the charter; **CC4** for its slice	Principal accepts accountability; Board for capability write charters; overseer confirms oversight/escalation
G3	Approved → Provisioned	Derived needs emitted; Grant Reconciliation Ledger shows no unaccepted difference; **WF3** holds	Platform team confirms provisioning
G4	Provisioned → Active	Envelope entries approved (**CC1**, **CC2**, **CC5**); agent identity bound	Approvers commit (O3)
G5	Active → Under review	Review date reached, or B2 evidence exceeds a trigger (reversal rate, denial rate), or a defect appears involving the charter	Automatic
G6	Under review / Degraded → Approved (re-approval)	As G2, plus evidence record reviewed	Charter Review; Board for level changes
—	Under review → Degraded	Review date passed without re-approval (**WF10** Stale); level lowered by one in M_t	Automatic
G7	Active / Degraded → Suspended	Incident; **CC6** unattributed evidence naming the agent; principal vacancy (**WF1**)	Overseer or agency architect; immediate
G8	Suspended → Retired	Envelope entries revoked; grants removed; needs recomputed for dependants	Charter Review

18.6 Invariants and conditions as gates

Every invariant of the execution plane and every condition of the design plane is also a gate. **Inv-1**–**Inv-3** are write-time rejections of the substrate: an approval by a non-human, an authorising

Table 18.4. Gates and states added or extended in v1.1 (continuation of Table 18.3); all SHOULD when the profile is in use

Gate	Transition	Conditions (tool)	Decisions (role)
G2	Analysed → Approved	additionally **WF13** (REVIEWED recorded), **WF15**, **WF16**, **WF17** (blocking, no acceptance), **WF18**, **WF19**	Charter Review records the reviewed configuration digest and the class bound
G4	Approved → Provisioned	**CC7**: a certificate for every envelope entry when the certificate profile is in use	Provisioning pipeline emits certificates with the envelope
G2′	Active → Under review	CONFIGDRIFT(a, m) raised by O6	Charter Review re-evaluates or rolls back; envelope entries flagged, no new mandate from m
G7	Active → Suspended	additionally BLASTEXCEEDED after an incident; UNCHARTED instance stopped	Agent owner; incident process
G8	Suspended/Active → Retired	check that no INSTANCE of the type remains (**WF14**)	Agent owner

observed envelope assertion, or a self-sourced envelope assertion is refused, not logged as a defect. **WF1**–**WF12** and **CC1**–**CC6** are evaluated at G2, G3, G4 and G6 and their failure blocks the transition unless accepted. **CC4** blocks the conferral of write authority on a slice below execution Level 3. **X-Inv-1**–**X-Inv-2** are write-time rejections of the boundary registers.

18.7 The Accepted-Defect Register

A defect the organisation chooses to live with is recorded, never silenced. Each entry names the defect and its witnesses, the business rationale, the accepting role (per Table 18.2), the compensating control if any, the owner and a review date. An accepted defect past its review date is reported again as unaccepted. The register is part of the Conformance Statement, so that a claim of conformance is always read together with the list of what it excludes. Acceptance of **WF1**, **WF4**, **WF8** and **CC5** defects is reserved to the Board because they are the conditions a regulator or an auditor will ask about first.

18.8 Evidence-gated promotion

New in Charterion v1.0

The level of a charter is a governance decision informed by evidence, not a configuration parameter. The rule below, which develops the business-case rule of El Hassani (2026f) that autonomy is *raised only at a lifecycle gate and lowered at any time*, makes the evidence explicit.

Evidence-gated promotion rule

A charter's level SHALL be raised only at gate G6, only by one step in $\mathbb{L}$, and only if over the review window its Evidence Record satisfies the organisation's thresholds on: (a) the reversal rate of permitted actions; (b) the denial rate; (c) the share of unwarranted outcomes; (d) the acceptance rate of its escalations by the overseer; (e) the absence of **CC6** unattributed evidence naming the agent. Promotion to *autonomous* additionally requires OVERSIGHT and ESCALATION to be current with $\mathrm{Span}(h) \leq \beta$ for the humans named. A charter's level MAY be lowered at any time by the agency architect or the overseer, and SHALL be lowered by the tool when the charter is stale (**WF10**). Thresholds are set by the Board and published in the Conformance Statement; a default set is proposed in Table 18.5.

Table 18.5. Default promotion thresholds (to be set by the Board; informative)

Evidence measure over the review window	propose → commit	commit → autonomous	Rationale
Reversal rate of permitted actions	$\leq 2\,\%$	$\leq 0.5\,\%$	An action the business undoes is an action the charter should not have permitted
Denial rate	$\leq 10\,\%$	$\leq 5\,\%$	Frequent denials mean the agent attempts what its scope does not cover
Unwarranted share	$\leq 5\,\%$	$\leq 1\,\%$	The slice is under-observed for the level sought
Escalation acceptance rate	$\geq 90\,\%$	$\geq 98\,\%$	Overseers agree with what the agent proposed at the boundary of its authority
Minimum window	90 days, 500 decisions	180 days, 2000 decisions	Enough evidence to estimate the rates

18.9 Assurance and replay

Because every decision is logged with its supporting assertions (**Inv-5**) and every envelope entry carries the charter identifiers that ground it (**CC1**), any action can be *replayed*: the auditor reconstructs the substrate as it was at the time, re-evaluates the admissibility function, and follows the provenance from the envelope entry to the charter, its principal and its rationale record. Theorem 11.1 guarantees that the chain exists for every permitted or escalated action under **CC1** and **CC5**. The Conformance Statement SHALL state the retention period of the decision log, which SHOULD not be shorter than the longest review window plus the regulatory retention applicable to the slice.

19 Regulatory and standards alignment

Charterion is not a compliance framework, but its constructs were chosen so that the obligations regulators place on organisations deploying AI can be *shown from the model*. Table 19.1 maps the principal instruments to the constructs that evidence them. The mapping is informative and does not constitute legal advice; obligations depend on the classification of the system and the jurisdiction.

Table 19.1. Alignment of Charterion constructs with regulatory and standards instruments (informative)

Instrument	Obligation or requirement	Charterion construct that evidences it
EU AI Act, Regulation (EU) 2024/1689 as amended by Regulation (EU) 2026/1744 (European Parliament and Council, 2024; European Union, 2026)	Art. 9 risk management; Art. 12 record-keeping (logging); Art. 13 transparency to deployers; Art. 14 human oversight (ability to understand, monitor, intervene, halt); Art. 26 deployer obligations (assign oversight to competent persons; monitor operation; keep logs). Annex III high-risk obligations apply from 2 December 2027 (stand-alone) and 2 August 2028 (Annex I embedded); Art. 50 transparency from 2 August 2026	Decision Log with supporting assertions (Art. 12); Oversight, Escalation, **WF7**, **WF9**, span bound (Art. 14, Art. 26); Reserved and **WF8** for decisions reserved to people; Evidence Records and B2 for post-market monitoring; Conformance Statement as documentation
ISO/IEC 42001:2023 AI management system (ISO/IEC, 2023)	Clauses 6 (planning, risk and impact assessment), 8 (operational controls, including Annex A controls on AI system lifecycle, data, roles and responsibilities, human oversight), 9 (performance evaluation), 10 (improvement)	Charter lifecycle and gates (8); roles and decision rights (Annex A); Agency Debt Dashboard and metrics (9); B2 refine and evidence-gated promotion (10)
NIST AI Risk Management Framework 1.0 (NIST, 2023)	Functions Govern (accountability structures, roles), Map (context, actors, impacts), Measure (metrics, monitoring), Manage (risk treatment, response)	Governance model (Govern); design-plane model with agents as principals and blast radius (Map); Chapter 20 (Measure); D4 resolutions and lifecycle gates (Manage)
NIST SP 800-207 Zero Trust Architecture (Rose et al., 2020)	Per-request access decisions; least privilege; policy engine and administrator; continuous monitoring	ADM as policy decision point over the substrate; **WF3** and D5 as least privilege computed from need; O6 drift detection
NIST NCCoE agent identity and authorization initiative (NIST NCCoE, 2026)	Identification of agents in the enterprise, delegation on behalf of humans, traceability of authority to human authorisation	Agent, Charter, Delegation, Acc; Theorem 11.1
IMDA Model AI Governance Framework for Agentic AI (IMDA, 2026)	Bounded autonomy; meaningful human checkpoints; accountability of deployers; technical controls and monitoring	Action levels and charters; Reserved, **WF8**, **WF9**; principals and Acc; Decision Log
Financial-services conduct rules (segregation of duties, four-eyes)	No single actor initiates and releases a payment or approves its own work	SoD, **WF4**; Reserved on release steps; caps and escalate bands

New in Charterion v1.1

From a mapping to computed evidence. Table 19.1 is a prose alignment: it names, for each instrument, the construct that would evidence it. With the obligation profile (Section 8.15) the alignment becomes data—Obligation$(o, \sigma, instr, kind)$ facts recorded by the regulatory owner—and the evidence becomes an output of the reasoner: for every human-decision obligation, the steps in scope with their reservation, staffing and the agents holding authority on them and UnmetObligation; for every record obligation, the decision-log records on the resources of the steps in scope with their **CC6** attribution and outcome shares; for every oversight obligation, the humans reachable from the agents active in scope with **WF9** and **WF14** span. The Regulatory Evidence Report (Appendix M; `regulatory_evidence.py`) is that output, per obligation, with a verdict of whether the model satisfies it. What remains prose—and remains the organisation's—is the reading of a statute into obligations and the judgement of whether computed evidence satisfies an article. Table 19.1 is retained as the informative starting point for that reading.

20 Metrics

Charterion defines three families of metrics, all computed from the model and the registers, so that they are reproducible and comparable across slices and over time. Definitions are normative; targets are set by the organisation.

20.1 Design-time metrics

Defect counts by condition
: the cardinality of each defect predicate: Orphan, Uncovered, Shadow, Missing, SoDViol, Contention, OverDeleg, Unguarded, Intrusion, Unreachable, Overloaded, Unbounded, Expired, Stale, Collusion, Uncompensated — reported as unaccepted and accepted.

Latent write authority
: $\mathrm{AD}_{\mathrm{latent}} = |\{(a,s) : \textsc{Auth}(a,s,\ell),\ \ell \geq \mathit{commit},\ \neg\textsc{Agentic}(s)\}|$ — write authority over steps the business has not designated for agents.

Agency debt
: **New in Charterion v1.0**
The *agency debt* of a slice is the vector
$$\mathrm{AD} = (\mathrm{AD}_{\mathrm{latent}}, \mathrm{AD}_{\mathrm{reserved}}, \mathrm{AD}_{\mathrm{shadow}}, \mathrm{AD}_{\mathrm{contention}}, \mathrm{AD}_{\mathrm{orphan}}, \mathrm{AD}_{\mathrm{unwatched}}, \mathrm{AD}_{\mathrm{stale}})$$
with $\mathrm{AD}_{\mathrm{reserved}} = |\textsc{Intrusion}|$, $\mathrm{AD}_{\mathrm{shadow}} = |\textsc{Shadow}|$, $\mathrm{AD}_{\mathrm{contention}} = |\textsc{Contention}|$, $\mathrm{AD}_{\mathrm{orphan}} = |\textsc{Orphan}|$, $\mathrm{AD}_{\mathrm{unwatched}} = |\textsc{Unguarded}|$, $\mathrm{AD}_{\mathrm{stale}} = |\textsc{Stale}| + |\textsc{Expired}|$. The *Agency Debt Index* is $\mathrm{ADI} = \sum_i w_i\,\mathrm{AD}_i \,/\, |\textsc{Agentic}|$ with weights w_i published in the Conformance Statement (equal weights by default). Agency debt is the agentic analogue of technical debt: authority conferred beyond what is justified, coordinated or watched, accumulating silently until it is paid in incidents. It is reported per slice and trended; the cases of Part VII report it for each iteration.

Reach and blast radius
: $|\mathrm{Reach}(a)|$ and $|\mathrm{Blast}(a)|$ per agent (Section 8.7); the maximum blast radius in a slice.

Charter granularity
: distribution of write-level charters by scope sort (capability, process, step).

Oversight span
: $\mathrm{Span}(h)$ per human; maximum and mean in a slice; count of humans above β.

Coherence
: |UngroundedEnvelope|, |UnservedNeed|, |OverMandate|, |LevelMismatch|, |BrokenChain|.

20.2 Run-time metrics

Per charter and per slice over a window: counts and shares of permit, deny, escalate and unwarranted outcomes; reversal rate of permitted actions; escalation acceptance rate; median time-to-decision for escalations; share of decisions whose minimum warrant $\kappa_{\min}$ falls in each band; number of **CC6** unattributed records; assertion currency (share of assertions in the acted-upon slice re-observed within their validity); drift events per source. The three agent KPIs of the business case (El Hassani, 2026f)—*admissible-action rate*, *escalation rate* and *reversal rate*—are the headline subset.

20.3 Boundary metrics

Per relationship: exposure level; number of current mandates and their band utilisation; ADM$^{\times}$ outcome shares; commitments active, satisfied, violated, cancelled; trust weight $t_A(B)$ and its trajectory; disputed or reversed cross-boundary actions (testable proposition XP1 of the fourth article predicts these fall with exposure level).

New in Charterion v1.1

Metrics added in v1.1 (informative). Design time: $\mathrm{AD}_{\mathrm{drift}} = |\textsc{ConfigDrift}|$, $\mathrm{AD}_{\mathrm{unstaffed}} = |\textsc{UnstaffedReserved}|$, $\mathrm{AD}_{\mathrm{amplified}} = |\textsc{CapAmplification}|$, $\mathrm{AD}_{\mathrm{obligation}} = |\textsc{UnmetObligation}|$, $\mathrm{AD}_{\mathrm{perimeter}} = |\textsc{OutOfPerimeter}| + |\textsc{DepthExceeded}|$, $\mathrm{AD}_{\mathrm{blast}} = |\textsc{BlastExceeded}|$, reported *beside* the seven v1.0 components; the v1.0 Agency Debt Index is unchanged (its definition is an existing algorithm under Appendix H), and an informative $\mathrm{ADI}_{1.1}$ = (sum of the six v1.1 components)/(agentic steps) is reported next to it (Table 23.14). Capacity: $\textsc{ReservedLoad}(h)$ and $\textsc{SpanI}(h)$ beside the oversight span; instances per type. Run time: certificate validity rate = |valid|/|approved entries| and time-to-verify per decision (AP17). Boundary: the trust estimate $t_A(B)$ of Appendix O with its effective sample size.

Part V

Artefacts and viewpoints

21 The artefact framework

21.1 Principles of the artefact framework

Charterion artefacts follow three rules that distinguish them from the deliverables of classical frameworks. First, every artefact is a *projection of the model or of a register*: it is generated, never authored, so it cannot disagree with the model and it is current whenever the model is. Second, every artefact has a *formal source*: the predicate or register whose content it presents, stated in its definition, so that two tools produce the same artefact from the same model. Third, artefacts are organised in the TOGAF manner—catalogues, matrices, diagrams (viewpoints) and reports—so that an architecture team can place them in its existing repository and Architecture Content Framework without translation.

21.2 Catalogues

21.3 Matrices

Authority Matrix
: agents × steps, cell = derived level (Auth); reserved steps marked; write-level cells on non-agentic steps highlighted (latent authority); intrusion cells flagged. Source: Auth, Agentic, Reserved.

Need-vs-Grant Matrix
: agents × resources, cell = (need level, grant level); shadow and missing cells flagged. Source: Need, Grant.

Contention Matrix
: per resource, agents × agents, cell = contention (unresolved), precedence (resolved, with direction) or none. Source: Contention, Precedence.

SoD Matrix
: SoD pairs × agents, cell = covers first, covers second, covers both (violation). Source: SoD, Covers.

Oversight Span Matrix
: humans × agents, cell = oversight, escalation, accountable; span per human against β. Source: Oversight, Esc, Acc.

21.4 Viewpoints

Viewpoints are defined in the manner of ArchiMate: stakeholders, concerns, elements and relations shown, and the formal source. All are generated from the least model. Table 21.2 defines the seven Charterion viewpoints.

21.5 Reports

Well-formedness Report
: per condition **WF1**–**WF12**: count, witnesses in business terms, accepted/unaccepted, change since last run. Generated by D3.

Table 21.1. Catalogues of Charterion

Catalogue	Content (fields)	Formal source	Owner; phase
Agent Register	agent id, platform identity, description, lifecycle state, slice	Agent	Agency architect; D2
Charter Register	charter id, agent, scope (sort and element), level, principal, state, validity, review, cap, rationale record ref.	Charter, Validity, Review, Cap	Agency architect; D2
Delegation Register	delegator, delegatee, scope, level ceiling, derived level (from Auth), validity	Delegation	Agency architect; D2
Guard Register	SoD pairs; precedences (agent, agent, resource); oversight (agent, human); escalation (agent, human); span per human	SoD, Precedence, Oversight, Escalation, Span	Agency architect, overseers; D2
Principal & Vacancy Register	roles, holders, vacancies, charters affected by each vacancy	Role, Holds, Charter	Agency architect; D1
Authority Request Register	request, agent, scope, level, justification (refused request or uncovered step), pre-analysis delta, decision, decider, timestamps	ARP (Chapter 16)	Principal, Board; D2 via ARP
Grant Reconciliation Ledger	per (agent, resource): provisioned level, derived need level, difference class (aligned, shadow, missing), disposition	Grant, Need, Shadow, Missing	Platform team; D5
Mandate Register	mandate id, agent, operations, counterparties, bands, validity, issuer, grounding charters	$\mathcal{M}_t$	Approvers; X2
Exposure Declaration	per relationship: exposure level, external affordance inventory, boundary envelope, cadence of trust re-estimation	X_B	Approvers; X1
Trust Weight Register	counterparty, $t_A(B)$, last re-estimation, commitment statistics	$t_A(\cdot)$	Substrate owner; X4
Assertion Store	the substrate $\mathcal{A}$ by layer with warrant attributes	$\mathcal{S}$	Substrate owner; O1–O3
Decision Log	request, outcome, supporting assertions, $\kappa_{\min}$, time, approver for approvals, attribution to charters	$\mathcal{L}$	Substrate owner; O5
Commitment Ledger	debtor, creditor, antecedent, consequent, state, shared referent	$\mathcal{C}$	Substrate owner; O5, X3

Table 21.2. Viewpoints of Charterion

Viewpoint	Stakeholders; concern	Shows	Formal source
Agency Layer Viewpoint	Architects; *what agents exist and what binds them*	Agents, charters (to scope and principal), delegations, guards, on top of the business layer	Agent, Charter, Delegation, guards, Realizes, Step
Accountability Chain Viewpoint	Risk, audit, regulators; *who answers for this agent*	For an agent: the chain from each write-level authority back through delegations to charters, principals and humans; vacancies highlighted	Auth, Delegation, Charter, Acc, Holds
Delegation Graph Viewpoint	Architects, security; *how authority flows between agents*	Directed graph of delegations with derived levels; over-delegations flagged; blast radius per node	Delegation, Auth, OverDeleg, Blast
Autonomy & Oversight Viewpoint	Overseers, Board; *who watches whom and how much*	Autonomous agents, their overseers and escalation humans, spans against β, reach and blast radius	Oversight, Esc, Span, Reach, Blast
Charter Scope Map	Business owners; *what an agent may touch*	For a charter: the steps in scope, marked agentic / reserved / undecided, with the resources and levels required; latent authority visible	InScope, Agentic, Reserved, Requires
Boundary Viewpoint	Approvers, commercial owners; *what we expose and what we send*	Per relationship: exposure level, exposed operations, mandates held by our agents, trust weight, active commitments	X_B, $\mathcal{M}_t$, $t_A(B)$, $\mathcal{C}$
Agency Grid Viewpoint	Everyone; *one page*	The grid of Table 5.1 instantiated for a slice: per cell, counts of elements, defects and conformance level	All

Table 21.3. Catalogue fields, matrices, viewpoints and reports added in v1.1 (continuation of Tables 21.1 and 21.2)

Artefact	Addition (fields)	Formal source	Owner; phase
Charter Register	`reviewed_config` (κ); `obligations` in scope	Reviewed; Obligation	Charter Review; G2
Agent Catalogue	`running_config`; `class`; `blast_bound`; *Instances* sub-table (id, type, since)	Config, AgentClass, BlastBound, Instance	Agent owner; O1
Principal & Vacancy Register	*Reserved steps* section: step, performer role, holders, load	Performs, UnstaffedReserved, ReservedLoad	HR / process owner; D1
Delegation Matrix	`declared` (collaboration edge), `permit_cap`, `unit`, validity	Collaboration, DelegationCap, delegation validity	Architect; D2
Envelope Register	`certificate`; `review_flag`	Definition 11.1; **WF13**	Pipeline; D5
Mandate Register	`certificate` of the grounding need; `delegation_cap`	Section 10.4; **WF16**	Boundary owner; X2
Decision Log	`certificate`	**CC7**	Runtime; O2
Configuration viewpoint (new)	agents × charters with reviewed and running digests, drift marked	**WF13**	Agent owner
Human-work viewpoint (new)	reserved steps × roles × holders with load, beside the oversight span	**WF15**, **WF14**	Process owner
Audit viewpoint (new)	decision → certificate → charter → human	**CC7**, Theorem 11.1	Auditor
Regulatory Evidence Report (new)	per obligation: instrument, kind, scope, computed evidence, verdict	**WF17**, Appendix M	Regulatory owner; O5
Well-formedness Report	lines for **WF13**–**WF19** and **CC7**; v1.1 agency-debt components	Table 8.3	Tool; D3

Coherence Report
: per condition **CC1–CC6**: as above. Generated by D3.

Agency Debt Dashboard
: the agency-debt vector and index per slice with trend; latent authority; spans; blast radii. Generated by D3.

Charter Rationale Record
: per write-level charter: business purpose, why this scope and level, why not narrower, principal's acceptance, compensating controls, date. Authored in D2; the one artefact that is written by people, because it records a decision.

Accepted-Defect Register
: see Chapter 18.

Evidence Record
: per charter: outcome counts and rates over the window, escalations and their decisions, reversals, drift events, **CC6** attributions. Generated by O5.

Conformance Statement
: see Section 25.4.

21.6 The Charterion Model Exchange Format

New in Charterion v1.0

Charterion-MXF is a JSON format for the interchange of Charterion models between repositories, reasoners and validators. It is the concrete syntax of the metamodel: each section carries the facts of one group of predicates, and a document is well typed if and only if it is a Charterion model. The schema is given in Appendix G and supplied with the toolkit.

A document has the sections `core` (business, application and provisioned-technology facts), `agency` (agents, charters, delegations, guards), `extensions` (reserved, escalation, validity, caps, span bound), `bridge` (grants and the envelope entries in force, with approver and status), `boundary` (mandates, exposures, trust weights), `parameters` ($\theta_{\mathrm{id}}, \theta_{\mathrm{act}}, \beta$) and `metadata`. Identifiers are strings; levels are the four literals; dates are ISO 8601. A conformant tool SHALL read and write the format without loss for the sections it supports and SHALL reject a document that violates referential integrity. Human-readable names, descriptions and diagram layout are carried as annotations and ignored by the semantics.

New in Charterion v1.1

Charterion-MXF 1.1. The exchange format gains, all optional, the sections of Table G.3 (Appendix G): carriers for IRREVERSIBLE and COMPENSATION (which v1.0 defined but could not exchange), validity on delegations, one section per v1.1 profile, certificates on envelope entries, a decision log, and a `conformance` section carrying the profiles in use, the per-slice execution levels and the per-relationship exposure levels that **CC4** needs. A 1.0 document is a valid 1.1 document; `charterion_mxf_version` is `"1.0"` or `"1.1"`. The JSON Schema is `charterion_mxf_schema_1_1.json`; the certificate and the three registers have their own schemas (Appendix L).

Part VI

Tooling

22 Reference implementation

22.1 Components

The reference implementation supplied with this version comprises: `charterion.py`, the core reasoner of the fifth article, which computes the least model of Figure 7.2 by semi-naïve fixpoint over Python sets and reports the seven core defects with witnesses; `charterion_ext.py`, the v1.0 extension analysis, which builds the time-indexed model M_t, evaluates **WF8**–**WF10** (the temporal profile on charters; its extension to delegations is specified in Chapter 8 but not yet implemented) and **CC1**–**CC3** and **CC5** (**CC4** and **CC6** require per-slice conformance levels and a decision log, which the reference implementation does not yet read), computes reach, blast radius, spans and agency debt; `charterion_separation.py` evaluates **WF11** and **WF12** on a results file or model and is the pre-analysis function used by the Authority Request Protocol when run on a model extended with a candidate charter; and renders the Well-formedness, Coherence and Agency Debt reports; `charterion_validate.py`, a command-line validator that checks an Charterion-MXF document against the schema and referential integrity, runs the analysis at a given instant and exits with a non-zero status when defects of the selected classes exist; `charterion_mxf_schema.json`, the exchange-format schema; the case models of Part VII; and a test suite. All components use only the Python standard library, so that they can be read as a specification and run anywhere. (*E16, v1.1: the `wf12` function of `charterion_separation.py` evaluates uncompensated autonomy on* NEED*, which is vacuous on the five cases; the rule D12 of Chapter 8 is on* AUTH *and is what `wf12_arp_demo.py` and the v1.1 module compute.*)

22.2 Repository integration pattern

Repository check pattern

The design plane SHOULD be kept under version control as Charterion-MXF documents, and the validator SHOULD run as a check on every proposed change (a pull request or its equivalent), failing the change when any unaccepted **WF** or **CC** defect appears. The Accepted-Defect Register is itself versioned alongside the model so that the check can distinguish accepted from new defects. Provisioning (D5) SHOULD be executed by the same pipeline from the merged model, so that identity management is never edited by hand for chartered agents.

This pattern is what makes the method continuous. It costs nothing that a software team does not already have, and it makes **WF3** and **CC1**–**CC2** hold by construction: the pipeline that provisions is the pipeline that verified.

22.3 Integration with runtime and identity systems

The reference implementation does not enforce; it computes what is to be enforced. Three interfaces connect it to the systems that do. *Identity management* receives the derived NEED set as the target entitlement state for each agent identity (via SCIM, the IdP's API or an infrastructure-as-code representation) and returns the provisioned state as GRANT facts. *The policy decision point or agent runtime* receives the charter-derived envelope as policy—for an attribute-based engine, one rule per envelope entry with the charter identifier as an attribute; for a runtime

such as AARM (Errico, 2026) or a gateway implementing the OWASP Agent Control Standard (OWASP GenAI Security Project, 2026), the envelope as the declarative policy those systems evaluate at the tool boundary—and returns its decision log as $\mathcal{L}$. *Observation sources* (CMDB, API catalogues, MCP servers, process engines) feed the anchor, affordance and coordination layers of the substrate. Charterion is agnostic about these systems; what it requires of them is stated in Chapter 24.

New in Charterion v1.1

Reference implementation, v1.1. The v1.0 modules are shipped byte-identical. Added: `charterion_v11.py` (stratum 5, **WF13**–**WF19**, **CC7**; **WF10** on delegations, **CC4** and **CC6**, which v1.0 specified but did not implement—Table 26.1 row "Coherence" is corrected accordingly; **WF11** and **WF12** computed on every run with the D12 rule of Chapter 8, on AUTH); `charterion_cert.py` (certificate builder and checker); `charterion_validate_v11.py` (MXF 1.1 validation wrapping the unchanged 1.0 validator); `regulatory_evidence.py`; `trust_estimator.py` (informative); `test_v11.py` (the conformance test suite of this version: witness and non-witness model per condition, vacuity, certificate soundness, completeness and tamper rejection, round-trip, Appendix H regression); `injection_study.py`, `regression_v11.py`; and `charterion_core.dl`, a Soufflé rendering of Figure 7.2 and stratum 5 delivered *untested*—no Soufflé binary was available in the authoring environment—and therefore not part of the executable definition of the version until validated against the Python reference. Standard library only.

Integration with EA repositories through MCP. Three EA tools now expose their repositories to agents through MCP servers (SAP LeanIX, generally available November 2025; Ardoq, September 2025; Bizzdesign, announced January 2026). In the pattern of Section 21.6 these servers are an *import* path for D1 (the core facts of a slice can be read from the repository by an agent) and, where the tool allows writes, an *export* path for the Charter Register; they are not a substrate: the agent acts under the invoking user's or the token's permissions, the repository is not consulted at decision time, and answers carry no warrant. A binding SHOULD keep the repository's own access control in force for the agent that populates the model and SHOULD route any write of a charter through gate G2 (informative; no condition is attached). Primary documentation retrieved 6 September 2026 (Appendix I).

22.4 Tool conformance

A tool claims conformance by running the test models supplied with the reference implementation and reproducing the reference least model and defect sets exactly; by validating and round-tripping the example Charterion-MXF documents; and by passing the extension test suite for the profiles it supports. The test suite and its expected outputs are part of the toolkit.

Part VII

Worked cases

23 Five sector cases

23.1 Purpose and method of the cases

The cases demonstrate the framework on five fictitious organisations in five sectors, each modelled at the level of detail of a first architecture pass, each run through two iterations of the design loop—a first chartering with broad scopes as an architecture team might produce it, and a refinement in which write-level charters are re-scoped to steps and guards are added—and each analysed by the reference implementation. Every number in this part is computed by the toolkit from the case models supplied with it (`cases/`), and the tables are generated from the result files; nothing is transcribed by hand. The cases are the author's and demonstrate that the constructs are operational and that the analyse–refine loop converges; they do not show that a real estate can be described without loss (Section 26.4).

The cases were chosen to exercise different conditions. Meridian Insurance (UC1) exercises all seven core conditions. Northgate Retail Bank (UC2) emphasises separation of duty, reserved regulatory decisions and the oversight of autonomous agents. Saint-Aubin University Hospital (UC3) emphasises broad read charters against narrow write charters and contention on a shared clinical record. The Regional Benefits Agency (UC4) emphasises accountability chains and the vacancies a reorganisation leaves behind. Helix Cloud Platform Operations (UC5) emphasises contention on shared infrastructure and autonomous remediation under oversight.

23.2 UC1 — Meridian Insurance

Meridian is the case of the fifth article (El Hassani, 2026e), whose structure follows the ArchiSurance case that accompanies ArchiMate (Lankhorst, 2017): five capabilities (claims handling, policy administration, customer service, fraud control, finance), each realised by one process; sixteen steps with their control-flow order, thirteen designated for agents (the claim decision, the referral to the special investigation unit and the posting of ledger entries are not); nine resources; five managers holding five roles and one vacant role. Nine agents are chartered on a first pass with broad scopes—an intake agent on the claims-handling capability at *commit*, a damage-assessment agent on the claims process at *propose*, a fraud-screening agent on the fraud-control capability at *commit* with a delegation from the intake agent, a customer-reply agent at *propose*, a renewal agent on policy administration at *commit*, a settlement agent on the claims process at *autonomous*, a reconciliation agent delegating to a ledger agent, and a legacy policy bot whose principal is the vacant role. Two SoD pairs, one oversight and one precedence are declared; grants are the derived needs minus one entitlement and plus two added by hand. The model has 168 facts.

The first iteration reports one orphan (the legacy bot), one uncovered step (send reply), two shadow grants and one missing grant, three SoD violations, eight contention pairs, one over-delegation and one unguarded autonomous agent, with 36 authority tuples, 27 coverage relations and 36 needs. Re-scoping the three write-level agents to steps removes two of the three SoD violations and half of the contention pairs without changing which agentic steps are covered (23 tuples, 17 covered steps, 25 needs); the four remaining contentions are genuine coordination questions resolved by one precedence declaration and by retiring the bot, which also resolves the orphan. In v1.0 terms, two of the three non-agentic steps—the claim decision

and the SIU referral—are declared Reserved (the ledger posting is left undecided to illustrate the third state); the broad write-level charters of iteration 1 produce three **WF8** intrusions that iteration 2 removes, and the vacant principal of the legacy bot is also a **WF9** unreachability. Table 23.1 lists the defects of both iterations, and Table 23.6 places Meridian alongside the four new cases.

Table 23.1. UC1 Meridian Insurance: well-formedness defects per condition (WF1–WF7) and reserved-step intrusions (WF8) in iteration 1 (broad charters) and iteration 2 (refined charters); counts in brackets, witnesses in words.

Condition	Iteration 1 (broad charters)	Iteration 2 (refined)
WF1 Accountable authority	[1] Legacy bot (principal Legacy Owner, vacant)	[1] Legacy bot (principal Legacy Owner, vacant)
WF2 Covered designation	[1] Send reply	[1] Send reply
WF3 Justified provisioning	[3] shadow: Damage assessment agent on Payment gateway (commit); Customer reply agent on Claims DB (read); missing: Renewal agent on Email service (commit)	[3] shadow: Damage assessment agent on Payment gateway (commit); Customer reply agent on Claims DB (read); missing: Renewal agent on Email service (commit)
WF4 Separation of duty	[3] Claim intake agent: Assess damage / Decide claim; Fraud screening agent: Check fraud indicators / Flag claim; Settlement agent: Assess damage / Decide claim	[1] Fraud screening agent: Check fraud indicators / Flag claim
WF5 Declared coordination	[8] 8 pairs: Email service (3), Claims DB (2), Policy DB (1), Payment gateway (1), General ledger (1)	[4] 4 pairs: Claims DB (2), Policy DB (1), Email service (1)
WF6 Bounded delegation	[1] Reconciliation agent to Ledger agent on Monthly close (commit declared; delegator holds propose)	[1] Reconciliation agent to Ledger agent on Monthly close (commit declared; delegator holds propose)
WF7 Watched autonomy	[1] Settlement agent	[1] Settlement agent
WF8 Reserved-step integrity	[3] Claim intake agent: Decide claim; Fraud screening agent: Refer to SIU; Settlement agent: Decide claim	[0] none

23.3 UC2 — Northgate Retail Bank

Northgate Retail Bank is a fictitious mid-sized retail bank. The modelled slice is the payments and KYC/AML domain: five capabilities (Payments, Customer onboarding, AML monitoring, Dispute handling, Treasury), each realised by one process. Regulation reserves four decisions to named humans—releasing a payment above the straight-through threshold, deciding whether to file a suspicious activity report (SAR), deciding to freeze an account, and ruling on a dispute—and the treasury policy reserves the placing of deposits to the Treasurer. The bank's agent platform hosts eight agents: five new ones built for this slice (payments, onboarding, screening, dispute, reconciliation), two analytical assistants (alert triage, liquidity forecast) and a legacy customer chatbot inherited from a digital-channels team that was dissolved in the 2025 reorganisation; its owning role (Digital Channels Owner) has been vacant since. The platform team provisioned entitlements from the derived needs of the first model, with two departures. The dispute agent never received access to the Dispute case system because it had been chartered at *propose* and the team read that as "no system access needed" (a **missing** grant at the propose level). The alert triage agent was provisioned from the screening agent's template and therefore received a *commit* entitlement on the Regulator portal it never needs (a **shadow** grant that would let a triage assistant file a regulatory report). The legacy chatbot keeps a *read* entitlement on the Payments ledger for balance look-ups; in iteration 2 this survives as the only shadow grant, recorded in the Accepted-Defect Register until the chatbot is decommissioned.

Iteration 1. The first-pass model has 210 facts: 8 agents, 26 steps (21 agentic, 5 reserved), 11 resources, 8 charters and 2 delegation(s). The analysis derives 48 authority tuples and 31 needs and reports 1 orphan(s), 2 uncovered step(s), 1 shadow and 1 missing grant(s), 4 separation-of-duty violation(s), 14 contention pair(s), 1 over-delegation(s) and 1 unguarded autonomous agent(s); the v1.0 extension reports 4 write authorities on human steps, of which 4 are intrusions on reserved steps (**WF8**). The Agency Debt Index is 1.190. The first pass gives every agent one charter on the capability or process it serves, at the highest level any of its steps might need. The consequences are

computed, not assumed. **WF8/WF4/WF7—payments.** The payments agent, chartered *autonomous* on the whole Payments capability for straight-through processing, thereby holds authority over *Release payment*, which is reserved to the Head of Payments above the threshold (**WF8** intrusion), covers both *Initiate transfer* and *Release payment* (**WF4**: the four-eyes rule of payments), and runs without oversight (**WF7**). **WF4—onboarding.** The onboarding agent's capability-wide *commit* charter covers both *Screen customer* and *Open account*: the agent that screens would also decide (**WF4**). **WF8/WF4/WF5—screening.** The screening agent's *commit* charter on AML monitoring reaches *Decide SAR* and *Decide freeze*, both reserved to the MLRO (two **WF8** intrusions), and covers *Investigate alert* together with *File SAR* (**WF4**). Its *Freeze account* step writes the Customer master and the Payments ledger, which the onboarding and payments agents also write from other processes, so three undeclared contentions arise (**WF5**). **WF2/WF6—disputes.** The architecture team was reluctant to give a customer-facing agent write authority and chartered the dispute agent at *propose*; but *Register dispute* and *Refund customer* require *commit*, so both designated steps are uncovered (**WF2**). The team's workaround—a delegation "dispute agent asks payments agent to refund" declared at *commit*—exceeds the delegator's own *propose* authority (**WF6**); by attenuation it delegates nothing useful. **WF8/WF5—treasury.** The reconciliation agent's process-wide *commit* charter covers *Place deposit*, reserved to the Treasurer (**WF8**), and because that step writes the Payment rail it also produces a spurious contention with the payments agent (**WF5**). **WF1/WF4/WF5—legacy chatbot.** The chatbot still carries a *commit* charter on Customer onboarding whose principal role is vacant: write-level authority with nobody accountable (**WF1**). Because it duplicates the onboarding agent's scope it also violates the screen/open SoD (**WF4**) and contends with the onboarding, screening and payments agents on Customer master, KYC document store, Payments ledger and Notification service (**WF5**). In total 14 contention pairs: six on the Payments ledger, three each on Customer master and Notification service, one each on KYC document store and Payment rail. **WF3.** One shadow and one missing grant, explained above. Latent write authority equals the four intrusions; the Agency Debt Index is 1.190 per agentic step.

Iteration 2. Iteration 2 keeps broad *read* charters and re-scopes every write-level charter to a step. The payments agent keeps *commit* on *Capture instruction* and *Confirm to customer* and *autonomous* only on *Initiate transfer* (below-threshold straight-through processing), now under the oversight of the Head of Payments (**WF7** resolved); *Release payment* is chartered to nobody (**WF8** and **WF4** resolved). *Screen payment* moves to the screening agent so that screening is independent of initiation. The onboarding agent holds *commit* on *Collect documents*, *Verify identity* and *Open account* and *propose* on *Assess risk*; *Screen customer* is chartered to the screening agent directly, so the iteration-1 delegation is withdrawn and the screen/open SoD holds. The screening agent's write charters are limited to *Score transactions*, *Raise alert*, *File SAR* and *Freeze account*—the last two execute decisions the MLRO has already taken in the reserved steps—while *Investigate alert* is covered only by the alert triage agent at *propose* (investigate/file SoD resolved). The dispute agent receives *commit* on *Register dispute* and *Refund customer* (the refund executes the human ruling) and *propose* on evidence gathering and resolution proposals; the **WF6** delegation is withdrawn and the Dispute case system grant is provisioned (**WF2**, **WF6** and the missing grant resolved). The reconciliation agent keeps *commit* on *Reconcile nostro* and *Post journal* only; with *Place deposit* back in human hands the Payment-rail contention disappears. The legacy chatbot is downgraded to *read* on Customer onboarding under the Head of Onboarding pending decommissioning (**WF1**, **WF4** and its contentions resolved). Five precedence declarations settle the remaining shared writes: the screening agent's freeze wins over the payments agent (Payments ledger) and over the onboarding agent (Customer master, Payments ledger); account creation precedes any transfer (onboarding before payments on the Payments ledger); transfers precede refunds on the Payment rail. Two defects are knowingly accepted and recorded: the chatbot's read grant on the Payments ledger (shadow, expires with decommissioning) and the payments/onboarding contention on the Notification service (an append-only queue for which ordering is immaterial). Every active agent has a declared escalation human; oversight spans are at most 2, far below the default bound of 12. All write-level charters carry validity 2026-10-01 to 2027-09-30 with review on 2027-03-31, except the autonomous transfer charter and the SAR-filing, freeze and refund charters, which are valid to 2027-03-31 with review on 2027-01-31. The Agency Debt Index falls from 1.190 to 0.095.

After refinement the model has 222 facts and 24 charters; the analysis derives 40 authority tuples covering 21 of 21 agentic steps with 23 needs, and reports 2 defect(s) in total (1 shadow grant(s), 1 contention pair(s), 0 other), no latent write authority and no intrusion. The Agency Debt Index falls from 1.190 to 0.095. Table 23.2 lists the defects of both iterations with their witnesses.

Table 23.2. UC2 Northgate Retail Bank: well-formedness defects per condition (WF1–WF7) and reserved-step intrusions (WF8) in iteration 1 (broad charters) and iteration 2 (refined charters); counts in brackets, witnesses in words.

Condition	Iteration 1 (broad charters)	Iteration 2 (refined)
WF1 Accountable authority	[1] Legacy chatbot (principal Digital Channels Owner, vacant)	[0] none
WF2 Covered designation	[2] Register dispute; Refund customer	[0] none
WF3 Justified provisioning	[2] shadow: Alert triage agent on Regulator portal (commit); missing: Dispute agent on Dispute case system (propose)	[1] shadow: Legacy chatbot on Payments ledger (read)
WF4 Separation of duty	[4] Payments agent: Initiate transfer / Release payment; Onboarding agent: Screen customer / Open account; Screening agent: Investigate alert / File SAR; Legacy chatbot: Screen customer / Open account	[0] none
WF5 Declared coordination	[14] 14 pairs: Payments ledger (6), Customer master (3), Notification service (3), KYC document store (1), Payment rail (1)	[1] Payments agent vs Onboarding agent on Notification service
WF6 Bounded delegation	[1] Dispute agent to Payments agent on Resolve dispute (commit declared; delegator holds propose)	[0] none
WF7 Watched autonomy	[1] Payments agent	[0] none
WF8 Reserved-step integrity	[4] Payments agent: Release payment; Screening agent: Decide SAR, Decide freeze; Reconciliation agent: Place deposit	[0] none

23.4 UC3 — Saint-Aubin University Hospital

Saint-Aubin University Hospital is a fictitious teaching hospital. The slice covers five capabilities—Appointment scheduling, Admissions, Clinical documentation, Pharmacy and Billing—realised by one process each, 23 steps in total. Five steps are clinical or professional decisions reserved to humans (validating a triage summary, signing a note, prescribing, deciding discharge, verifying a prescription); one step, coding validation, is non-agentic but has not yet been decided either way. The Electronic health record (EHR) is the shared resource of the slice: it is read by almost every step and written by admissions, clinical and pharmacy steps. Eight agents are hosted: scheduling, triage-summary drafting, coding and billing, pharmacy stock, patient communications, admissions, discharge drafting and claim submission. The Clinical Informatics Lead post is vacant after the incumbent left; the triage-summary agent had been chartered under it.

Entitlements were provisioned from the derived needs of the first model with three departures. Finance refused a *commit* entitlement on the General ledger to "a bot", so the coding and billing agent's derived need is unmet (**missing**). The patient communications agent received full *commit* on the EHR so that it could "personalise messages", although no step it covers writes the EHR (**shadow**). The scheduling agent's service account was copied from a clinical template and carries *read* on the e-prescribing system (**shadow**). Iteration 2 provisions exactly the derived needs; the general-ledger grant is issued once the posting step is chartered explicitly and placed under oversight.

Iteration 1. The first-pass model has 180 facts: 8 agents, 23 steps (17 agentic, 5 reserved), 11 resources, 8 charters and 1 delegation(s). The analysis derives 42 authority tuples and 25 needs and reports 1 orphan(s), 1 uncovered step(s), 2 shadow and 1 missing grant(s), 4 separation-of-duty violation(s), 9 contention pair(s), 1 over-delegation(s) and 1 unguarded autonomous agent(s); the v1.0 extension reports 5 write authorities on human steps, of which 3 are intrusions on reserved steps (**WF8**). The Agency Debt Index is 1.235. The first pass gives each agent a single broad charter. **WF1/WF8/WF4/WF5—triage summary.** The triage-summary agent is chartered *commit* on the Admissions capability under the vacant Clinical Informatics Lead role, so its write-level authority has no accountable human (**WF1**). The charter also reaches *Validate triage*, a clinician's decision (**WF8**), and covers both *Draft triage summary* and *Validate triage* (**WF4**). The same broad scope makes it a second writer of the admission record and the Bed board alongside the admissions agent (**WF5**). **WF8/WF4—admissions.** The admissions agent's process-wide *commit* charter likewise reaches *Validate triage* (**WF8**) and covers the draft/validate pair (**WF4**). **WF7/WF4/WF5—billing.** The coding and billing agent is chartered *autonomous* on Billing for end-to-end revenue-cycle automation, with no oversight (**WF7**); it covers both *Assign codes* and *Submit claim*, which compliance separates (**WF4**). A second agent, claim submission, was chartered *commit* on the same capability, so the two contend on the Billing system, the Coding system

and the General ledger (**WF5**) and the claim-submission agent also violates the coding/submission SoD (**WF4**). Both hold latent write authority over the undecided step *Validate coding*. **WF8—pharmacy.** The pharmacy stock agent's capability-wide *commit* reaches *Verify prescription*, a pharmacist's decision (**WF8**); its medication-administration writes on the EHR contend with the admissions and triage-summary agents (**WF5**). **WF5—scheduling.** The scheduling agent and the patient communications agent were both chartered *commit* on Appointment scheduling, so they contend on the Scheduling system and the Messaging gateway (**WF5**). **WF2/WF6—discharge letters.** The discharge-draft agent is chartered *propose* on Clinical documentation (a sensible level for drafting) but *Send discharge letter* requires *commit* on the Messaging gateway, so the step is uncovered (**WF2**). The team declared a delegation "send the letter for me" from the discharge-draft agent to the communications agent at *commit*, above the delegator's *propose* (**WF6**); attenuation reduces it to *propose* and the step stays uncovered. **WF3.** One missing and two shadow grants, explained above. Nine contention pairs in total, three of them on the EHR; five latent write authorities of which three are intrusions; Agency Debt Index 1.235.

Iteration 2. Iteration 2 applies the hospital's rule "look widely, act narrowly": every agent keeps a *read* charter on its capability or process, and write authority is chartered step by step. The triage-summary agent is re-pointed to the Chief Medical Officer and holds only *propose* on *Draft triage summary* (**WF1**, **WF8**, **WF4** resolved); the admissions agent holds *commit* on *Register admission* and *Assign bed* only. The coding and billing agent holds *propose* on *Assign codes* and *autonomous* on *Post payment*—a reversible cash-posting step—under the oversight of the Revenue Cycle Manager (**WF7** resolved); *Submit claim* is chartered to the claim-submission agent alone, so the coding/submission SoD holds and the two agents no longer share writes. The pharmacy stock agent holds *propose* on *Check interactions* and *commit* on *Reorder stock* and *Dispense to ward*; *Verify prescription* remains the pharmacist's. The scheduling agent holds *propose* on *Propose slot* and *commit* on *Confirm booking*; all outbound messages (*Send reminder*, *Send discharge letter*) are chartered *commit* to the patient communications agent, whose read charters span Appointment scheduling and Clinical documentation, so the delegation is withdrawn (**WF2** and **WF6** resolved). The only remaining pair of write-level agents on the EHR—admissions (admission record) and pharmacy stock (medication administration record)—is ordered by a precedence declaration: the admission record comes first. Grants are provisioned exactly from the derived needs (**WF3** resolved). Every agent has an escalation human; the largest oversight spans are 2 (Chief Medical Officer, Revenue Cycle Manager, Patient Access Manager). All write-level charters are valid 2026-11-01 to 2027-10-31 with review on 2027-04-30, except the autonomous payment-posting charter, valid to 2027-04-30 with review on 2027-02-28. The refined model is well formed: the Agency Debt Index falls from 1.235 to 0.000.

After refinement the model has 191 facts and 23 charters; the analysis derives 39 authority tuples covering 17 of 17 agentic steps with 21 needs, and reports 0 defect(s) in total (0 shadow grant(s), 0 contention pair(s), 0 other), no latent write authority and no intrusion. The Agency Debt Index falls from 1.235 to 0.000. Table 23.3 lists the defects of both iterations with their witnesses.

Table 23.3. UC3 Saint-Aubin University Hospital: well-formedness defects per condition (WF1–WF7) and reserved-step intrusions (WF8) in iteration 1 (broad charters) and iteration 2 (refined charters); counts in brackets, witnesses in words.

Condition	Iteration 1 (broad charters)	Iteration 2 (refined)
WF1 Accountable authority	[1] Triage-summary agent (principal Clinical Informatics Lead, vacant)	[0] none
WF2 Covered designation	[1] Send discharge letter	[0] none
WF3 Justified provisioning	[3] shadow: Scheduling agent on e-prescribing system (read); Patient communications agent on Electronic health record (commit); missing: Coding and billing agent on General ledger (commit)	[0] none
WF4 Separation of duty	[4] Triage-summary agent: Draft triage summary / Validate triage; Coding and billing agent: Assign codes / Submit claim; Admissions agent: Draft triage summary / Validate triage; Claim submission agent: Assign codes / Submit claim	[0] none
WF5 Declared coordination	[9] 9 pairs: Electronic health record (3), Billing system (1), General ledger (1), Scheduling system (1), Messaging gateway (1), Bed board (1), Coding system (1)	[0] none
WF6 Bounded delegation	[1] Discharge-draft agent to Patient communications agent on Document encounter (commit declared; delegator holds propose)	[0] none
WF7 Watched autonomy	[1] Coding and billing agent	[0] none
WF8 Reserved-step integrity	[3] Triage-summary agent: Validate triage; Pharmacy stock agent: Verify prescription; Admissions agent: Validate triage	[0] none

23.5 UC4 — Regional Benefits Agency

The Regional Benefits Agency is a fictitious public body that administers means-tested benefits. Six months before the analysis a reorganisation merged the Intake and Eligibility directorates into a single Operations directorate, abolished the casework team-lead grade and left the Head of Intake and Head of Eligibility posts unfilled; the charters written for the agent platform before the reorganisation still name those roles. The slice covers five capabilities—Claim intake, Eligibility assessment, Benefit payment, Appeals and Fraud review—with 20 steps, of which four are statutory decisions reserved to officers (deciding eligibility, approving a payment run, ruling on an appeal, referring a case to prosecution). Eight agents are hosted: intake, document chasing, eligibility, notification, payment, appeals, fraud screening and a casework assistant used by officers.

Entitlements were provisioned from the derived needs with three departures. The Appeals system belongs to another directorate and the access ticket for the appeals agent is still pending (**missing** at the propose level). The casework assistant was given *commit* on the Payment schedule to match a "schedule payment" tool that its vendor configuration exposes, although no step it covers writes that resource (**shadow**). The intake agent inherited the pre-reorganisation intake service account, which carried *read* on the Income registry that only eligibility steps need (**shadow**). Iteration 2 provisions exactly the derived needs.

Iteration 1. The first-pass model has 187 facts: 8 agents, 20 steps (16 agentic, 4 reserved), 11 resources, 8 charters and 2 delegation(s). The analysis derives 32 authority tuples and 33 needs and reports 4 orphan(s), 2 uncovered step(s), 2 shadow and 1 missing grant(s), 4 separation-of-duty violation(s), 14 contention pair(s), 1 over-delegation(s) and 1 unguarded autonomous agent(s); the v1.0 extension reports 4 write authorities on human steps, of which 4 are intrusions on reserved steps (**WF8**). The Agency Debt Index is 1.812. The first model is the pre-reorganisation charter set analysed against the post-reorganisation role holders. **WF1—vacancy.** The intake agent and the document-chaser agent are chartered *commit* under the Head of Intake, and the eligibility agent under the Head of Eligibility; both roles are vacant, so three agents hold write-level authority with no accountable human (**WF1**). A fourth orphan arises through delegation: the eligibility agent delegates *commit* on *Compute entitlement* to the casework assistant, whose own charter is only *propose* under the vacant Casework Team Lead; the assistant inherits write authority and inherits the eligibility agent's empty accountability set. **WF8/WF4—statutory decisions.** The eligibility agent's capability-wide *commit* reaches *Decide eligibility*, a statutory officer decision (**WF8**), and covers both *Compute entitlement* and *Decide eligibility* (**WF4**). The notification agent, chartered *commit* on the whole Eligibility assessment capability because

"it only sends letters", reaches the same decision (**WF8**, **WF4**). The payment agent, *autonomous* on Benefit payment, reaches *Approve payment run* (**WF8**) and covers schedule/approve (**WF4**). The fraud screening agent, *autonomous* on Fraud review, reaches *Refer to prosecution* (**WF8**), covers score/refer (**WF4**) and, unlike the payment agent, has no oversight (**WF7**). **WF6—delegating beyond authority.** The casework assistant declares a delegation to the payment agent on *Pay benefit* at *commit*—the officers' assistant "instructs" payment scheduling—but holds no authority whatsoever in that process (**WF6**); nothing flows, and **WF6** makes the void delegation visible. **WF2—appeals.** The appeals agent is chartered *propose* on Appeals, yet *Register appeal* and *Implement ruling* require *commit*; both designated steps are uncovered (**WF2**). **WF5—shared case record and correspondence.** Four agents (intake, document chaser, eligibility, notification) write the Case management system and the Correspondence service from different processes without precedence, giving twelve contention pairs; the intake and document-chaser agents also share the Document store, and the payment and fraud agents both write the Payment schedule (suspension versus scheduling)—14 pairs in total. **WF3.** Two shadow grants and one missing grant, explained above. Four latent write authorities, all intrusions; Agency Debt Index 1.812, the highest of the five cases, driven by the four orphans.

Iteration 2. Iteration 2 repairs the accountability chains first. The Director of Operations is recorded as acting holder of the Head of Intake role, so the intake and document-chaser charters regain an accountable human; the eligibility and casework-assistant charters are re-pointed to the Director of Operations role directly (**WF1** resolved by two different mechanisms). Both delegations are withdrawn (**WF6** resolved). Write authority is then re-scoped: the intake agent holds *commit* on *Receive claim* and *propose* on completeness checks and identity verification; the document chaser holds *commit* on *Request missing documents* only; the eligibility agent holds *read* on the capability and *propose* on *Compute entitlement*, leaving *Decide eligibility* to officers (**WF8** and **WF4** resolved); the notification agent holds *commit* on *Notify decision* only. The payment agent holds *commit* on *Schedule payment*, *propose* on *Reconcile returns* and *autonomous* on *Execute payment run*—executing a run an officer has approved—under oversight of the Head of Payments. The fraud screening agent holds *propose* on *Score risk*, *commit* on *Open fraud case* and *autonomous* on *Suspend payment* (time-critical) under oversight of the Fraud Review Lead (**WF7** resolved); *Refer to prosecution* stays with officers. The appeals agent receives *commit* on *Register appeal* and *Implement ruling* and *propose* on *Compile case file* (**WF2** resolved), and its Appeals-system access is provisioned (**WF3** resolved). Four precedence declarations order the Payment schedule (suspension over ruling implementation over routine scheduling) and the Correspondence service (document requests before decision letters). One contention is knowingly accepted and registered: the intake agent (new claims) and the appeals agent (ruled cases) both write the Case management system, but on disjoint records; the reasoner works at resource granularity, so the pair is recorded in the Accepted-Defect Register with that justification. The casework assistant keeps *propose* on Eligibility assessment under the Director of Operations; the Casework Team Lead and Head of Eligibility roles stay vacant but no longer anchor any charter (the Principal & Vacancy Register records them). Every agent has an escalation human; the Director of Operations now spans 5 agents (bound 12)—the price of acting arrangements, flagged for the Charter Review. Write-level charters under the acting Head of Intake and the two autonomous charters carry a short validity (2026-09-15 to 2027-03-14, review 2026-12-15); the others run to 2027-09-14 with review on 2027-03-15. The Agency Debt Index falls from 1.812 to 0.062.

After refinement the model has 194 facts and 18 charters; the analysis derives 23 authority tuples covering 16 of 16 agentic steps with 27 needs, and reports 1 defect(s) in total (0 shadow grant(s), 1 contention pair(s), 0 other), no latent write authority and no intrusion. The Agency Debt Index falls from 1.812 to 0.062. Table 23.4 lists the defects of both iterations with their witnesses.

Table 23.4. UC4 Regional Benefits Agency: well-formedness defects per condition (WF1–WF7) and reserved-step intrusions (WF8) in iteration 1 (broad charters) and iteration 2 (refined charters); counts in brackets, witnesses in words.

Condition	Iteration 1 (broad charters)	Iteration 2 (refined)
WF1 Accountable authority	[4] Intake agent (principal Head of Intake, vacant); Document-chaser agent (principal Head of Intake, vacant); Eligibility agent (principal Head of Eligibility, vacant); Casework assistant (principal Casework Team Lead, vacant)	[0] none
WF2 Covered designation	[2] Register appeal; Implement ruling	[0] none
WF3 Justified provisioning	[3] shadow: Intake agent on Income registry (read); Casework assistant on Payment schedule (commit); missing: Appeals agent on Appeals system (propose)	[0] none
WF4 Separation of duty	[4] Eligibility agent: Compute entitlement / Decide eligibility; Notification agent: Compute entitlement / Decide eligibility; Payment agent: Schedule payment / Approve payment run; Fraud screening agent: Score risk / Refer to prosecution	[0] none
WF5 Declared coordination	[14] 14 pairs: Case management system (6), Correspondence service (6), Document store (1), Payment schedule (1)	[1] Intake agent vs Appeals agent on Case management system
WF6 Bounded delegation	[1] Casework assistant to Payment agent on Pay benefit (commit declared; delegator holds no authority)	[0] none
WF7 Watched autonomy	[1] Fraud screening agent	[0] none
WF8 Reserved-step integrity	[4] Eligibility agent: Decide eligibility; Notification agent: Decide eligibility; Payment agent: Approve payment run; Fraud screening agent: Refer to prosecution	[0] none

23.6 UC5 — Helix Cloud Platform Operations

Helix Cloud Platform Operations is the fictitious SRE organisation of a cloud platform provider. The slice covers five capabilities—Incident response, Change management, Capacity management, Access provisioning and Cost optimisation—with 24 steps, 19 of them agentic. Five steps are reserved to humans: declaring a major incident, approving a change (the CAB), approving a capacity purchase, approving privileged access and approving a commitment purchase. Three infrastructure resources are shared across processes: the Cluster configuration (written by remediation, rollback, configuration change, scaling and rightsizing), the Deployment pipeline (rollback and change) and the IAM directory (grant, privileged approval, revocation). Eight agents are hosted: monitoring, remediation, change, provisioning, cost, capacity, rollback and a legacy cron bot that has run nightly rightsizing since before the platform team existed; its owning role, Capacity Planner, is vacant.
Entitlements were provisioned from the derived needs with three departures. The change agent never received the Deployment pipeline credential, so its derived *commit* need is unmet (**missing**). The monitoring agent was given *commit* on the Cluster configuration "in case it needs to restart a pod" although it covers no write step (**shadow**), and the cost agent's service principal was copied from the FinOps team's group membership and carries *read* on the IAM directory (**shadow**). Iteration 2 provisions exactly the derived needs.

Iteration 1. The first-pass model has 197 facts: 8 agents, 24 steps (19 agentic, 5 reserved), 11 resources, 8 charters and 1 delegation(s). The analysis derives 42 authority tuples and 30 needs and reports 1 orphan(s), 1 uncovered step(s), 2 shadow and 1 missing grant(s), 5 separation-of-duty violation(s), 15 contention pair(s), 1 over-delegation(s) and 1 unguarded autonomous agent(s); the v1.0 extension reports 6 write authorities on human steps, of which 6 are intrusions on reserved steps (**WF8**). The Agency Debt Index is 1.632. The first pass charters each agent capability-wide at the maximum level any step might need. **WF7/WF8/WF4—self-healing.** The remediation agent is chartered *autonomous* on Incident response for a "self-healing platform" without oversight (**WF7**). The scope reaches *Declare major incident*, reserved to the incident commander (**WF8**), and covers both *Propose remediation* and *Apply remediation*, which the four-eyes rule on production separates (**WF4**). The rollback agent, chartered *commit* on the same process, duplicates both problems (**WF8**, **WF4**) and contends with the remediation agent on the Incident

tracker, Cluster configuration, Deployment pipeline and Paging service (**WF5**). **WF8/WF4—change.** The change agent's capability-wide *commit* reaches *Approve change*, the CAB's decision (**WF8**), and covers approve/apply (**WF4**). **WF8—access and cost.** The provisioning agent's *autonomous* charter reaches *Approve privileged access* (**WF8**)—it is under the Security Lead's oversight, so no **WF7**—and the cost agent's *commit* charter reaches *Approve commitment purchase* (**WF8**) and covers recommend/apply rightsizing (**WF4**). **WF1/WF8/WF4/WF5—legacy cron bot.** The cron bot's *commit* charter on Cost optimisation has a vacant principal (**WF1**), reaches the commitment-purchase approval (**WF8**), covers the recommend/apply pair (**WF4**) and contends with the cost agent on the Cluster configuration and the Cloud billing account (**WF5**). **WF6/WF2—capacity.** The capacity agent is chartered *propose* on Manage capacity; to let it scale, the team declared a delegation from the cost agent at *commit* on that process. The cost agent holds no authority in Manage capacity, so the delegation exceeds it (**WF6**) and, by attenuation, delegates nothing: *Scale cluster* remains uncovered (**WF2**). The team believed the delegation had solved the problem; the reasoner shows it had not. **WF5—shared infrastructure.** Five agents write the Cluster configuration from four processes; only one precedence (remediation over change) had been declared, leaving nine undeclared pairs on that resource, three on the Deployment pipeline and one each on the Incident tracker, Paging service and Cloud billing account—15 pairs. **WF3.** Two shadow grants and one missing grant, explained above. Six latent write authorities, all intrusions—the highest of the five cases; Agency Debt Index 1.632.

Iteration 2. Iteration 2 separates observation from action and declares who yields to whom on shared infrastructure. The monitoring agent keeps *read* on Incident response, *commit* on *Detect anomaly* and *propose* on *Diagnose* and *Propose remediation*; the remediation agent holds *autonomous* on *Apply remediation* alone, under the oversight of the Head of SRE (**WF7** resolved), plus *commit* on *Close incident*; the propose/apply SoD is satisfied because proposal and application are now different agents (**WF4** resolved), and *Declare major incident* is chartered to nobody (**WF8** resolved). The rollback agent holds *commit* on *Roll back* only. The change agent holds *commit* on *Raise change request* and *Apply configuration change* (applying what the CAB approved), *propose* on risk assessment and verification, and *read* on the capability; approval stays human. The provisioning agent holds *commit* on *Receive access request* and *Revoke expired access*, *autonomous* on *Grant access* for standard, policy-conformant requests under the Security Lead's oversight, and *read* on the capability; privileged approval stays human. Cluster sizing gets a single writer: the capacity agent holds *commit* on *Scale cluster* and *Apply rightsizing* (**WF2** resolved), while the cost agent only analyses and recommends (*propose*), which also satisfies the recommend/apply SoD; the **WF6** delegation is withdrawn. The legacy cron bot's charter is retired—the agent remains in the Agent Register with no authority (**WF1** resolved). The remaining concurrent writers of the Cluster configuration (remediation, rollback, change, capacity) are ordered by a six-entry precedence matrix—remediation over everything, rollback over change and capacity, change over capacity—and rollback precedes change on the Deployment pipeline; the monitoring and remediation writes on the Incident tracker are ordered by the process itself (*Detect anomaly* before *Apply remediation* before *Close incident*), so no declaration is needed there. Grants match needs exactly (**WF3** resolved). Every active agent has an escalation human; the Head of SRE spans 3 agents. Write-level charters run 2026-10-15 to 2027-10-14 with review on 2027-04-15; the two autonomous charters (remediation, standard access grants) are valid to 2027-04-14 with review on 2027-01-15. The refined model is well formed: the Agency Debt Index falls from 1.632 to 0.000.

After refinement the model has 206 facts and 22 charters; the analysis derives 28 authority tuples covering 19 of 19 agentic steps with 20 needs, and reports 0 defect(s) in total (0 shadow grant(s), 0 contention pair(s), 0 other), no latent write authority and no intrusion. The Agency Debt Index falls from 1.632 to 0.000. Table 23.5 lists the defects of both iterations with their witnesses.

Table 23.5. UC5 Helix Cloud Platform Operations: well-formedness defects per condition (WF1–WF7) and reserved-step intrusions (WF8) in iteration 1 (broad charters) and iteration 2 (refined charters); counts in brackets, witnesses in words.

Condition	Iteration 1 (broad charters)	Iteration 2 (refined)
WF1 Accountable authority	[1] Legacy cron bot (principal Capacity Planner, vacant)	[0] none
WF2 Covered designation	[1] Scale cluster	[0] none
WF3 Justified provisioning	[3] shadow: Monitoring agent on Cluster configuration (commit); Cost agent on IAM directory (read); missing: Change agent on Deployment pipeline (commit)	[0] none
WF4 Separation of duty	[5] Remediation agent: Propose remediation / Apply remediation; Change agent: Approve change / Apply configuration change; Cost agent: Recommend rightsizing / Apply rightsizing; Rollback agent: Propose remediation / Apply remediation; Legacy cron bot: Recommend rightsizing / Apply rightsizing	[0] none
WF5 Declared coordination	[15] 15 pairs: Cluster configuration (9), Deployment pipeline (3), Incident tracker (1), Paging service (1), Cloud billing account (1)	[0] none
WF6 Bounded delegation	[1] Cost agent to Capacity agent on Manage capacity (commit declared; delegator holds no authority)	[0] none
WF7 Watched autonomy	[1] Remediation agent	[0] none
WF8 Reserved-step integrity	[6] Remediation agent: Declare major incident; Change agent: Approve change; Provisioning agent: Approve privileged access; Cost agent: Approve commitment purchase; Rollback agent: Declare major incident; Legacy cron bot: Approve commitment purchase	[0] none

23.7 Cross-case reading

Table 23.6 and Table 23.7 place the five cases side by side. Four observations hold across all of them.

1. **The first pass is never well formed, and every finding is actionable.** Each iteration-1 model exhibits every core condition and **WF8**, and each witness names a business fact—a vacant role, a template-cloned grant, a capability charter at a write level, a delegation declared above the delegator's authority—that an architect can change. No finding is an artefact of the formalism.
2. **Re-scoping write-level charters to steps removes latent authority entirely and most contention.** Across the five cases latent write authority on human steps falls to zero in iteration 2, intrusions on reserved steps fall to zero, and contention pairs fall from 60 to 6 in total; the remaining pairs are genuine coordination questions that a precedence declaration or an accepted defect settles. Coverage of agentic steps is preserved or completed in every case.
3. **Agency debt is a usable summary.** The Agency Debt Index falls by between 56 % and 100 % from iteration 1 to iteration 2, and what remains is exactly the content of the Accepted-Defect Register (a legacy bot's read entitlement pending decommissioning, a notification-service contention accepted as harmless). The index is a fair one-number account of what the refine phase achieved and of what the organisation chose to keep.
4. **Orchestration voids separation of duty in three of five first passes. WF11** finds a collusion pair in Meridian (the intake agent delegates to the fraud-screening agent while both cover opposite sides of the assess/decide and screen/flag pairs), in Northgate (the onboarding agent delegates customer screening to the screening agent and itself opens the account) and in the Benefits Agency (the casework assistant delegates to the eligibility agent across the compute/decide pair); none is a **WF4** violation, since two different agents are involved, and all three disappear in iteration 2 when the write-level side is chartered directly under its own principal (Table 23.8). Shared-principal pairs remain in four cases and are reported for the Board's judgement rather than as defects.
5. **Vacancies are the recurring accountability failure.** All five organisations carry at least one charter whose principal is a role vacated by a reorganisation or a departure; in each the agent kept acting under authority nobody answered for. **WF1** finds it at design time; **WF9** would find the same agents unreachable for escalation; **WF10** would have

degraded their charters at the first missed review. The three conditions are the framework's answer to the most ordinary way in which agent authority becomes orphaned: not by attack, but by organisational change.

Table 23.6. Use-case summary per chartering iteration: model facts, agents, agentic steps (Ag. steps), authority tuples (Auth.), covered agentic steps, derived needs, latent write authority (Latent), reserved-step intrusions (Intr.), undeclared contention pairs (Cont.) and the Agency Debt Index (ADI, equal weights, per agentic step).

Case	It.	Facts	Agents	Ag. steps	Auth.	Covered	Needs	Latent	Intr.	Cont.	ADI
UC1 Meridian Insurance	1	168	9	13	36	14	36	3	3	8	1.385
UC1 Meridian Insurance	2	157	9	13	23	12	25	0	0	4	0.615
UC2 Northgate Retail Bank	1	210	8	21	48	23	31	4	4	14	1.190
UC2 Northgate Retail Bank	2	222	8	21	40	21	23	0	0	1	0.095
UC3 Saint-Aubin University Hospital	1	180	8	17	42	19	25	5	3	9	1.235
UC3 Saint-Aubin University Hospital	2	191	8	17	39	17	21	0	0	0	0.000
UC4 Regional Benefits Agency	1	187	8	16	32	17	33	4	4	14	1.812
UC4 Regional Benefits Agency	2	194	8	16	23	16	27	0	0	1	0.062
UC5 Helix Cloud Platform Operations	1	197	8	19	42	22	30	6	6	15	1.632
UC5 Helix Cloud Platform Operations	2	206	8	19	28	19	20	0	0	0	0.000

Table 23.7. Agency-debt components per case and iteration (counts: Latent = AD_{latent}, Reserved = AD_{reserved} (intrusions), Shadow = AD_{shadow}, Cont. = $AD_{\text{contention}}$, Orphan = AD_{orphan}, Unwatched = $AD_{\text{unwatched}}$) and the Agency Debt Index = sum of the six components divided by the number of agentic steps (equal weights).

Case	It.	Latent	Reserved	Shadow	Cont.	Orphan	Unwatched	Sum	Ag. steps	ADI
UC1 Meridian Insurance	1	3	3	2	8	1	1	18	13	1.385
UC1 Meridian Insurance	2	0	0	2	4	1	1	8	13	0.615
UC2 Northgate Retail Bank	1	4	4	1	14	1	1	25	21	1.190
UC2 Northgate Retail Bank	2	0	0	1	1	0	0	2	21	0.095
UC3 Saint-Aubin University Hospital	1	5	3	2	9	1	1	21	17	1.235
UC3 Saint-Aubin University Hospital	2	0	0	0	0	0	0	0	17	0.000
UC4 Regional Benefits Agency	1	4	4	2	14	4	1	29	16	1.812
UC4 Regional Benefits Agency	2	0	0	0	1	0	0	1	16	0.062
UC5 Helix Cloud Platform Operations	1	6	6	2	15	1	1	31	19	1.632
UC5 Helix Cloud Platform Operations	2	0	0	0	0	0	0	0	19	0.000

Superseded in v1.1 by Table 23.14, which adds the AD_{stale} *column (E5) and the v1.1 components; the six components above are unchanged.*

Table 23.8. **WF11** independent separation on the five cases: collusion pairs (counts in brackets; witnesses as agent pair, separation-of-duty pair and kind of dependence) and shared-principal pairs (informative), per iteration; computed by `charterion_separation.py` from `cases_results.json`.

Case	Iteration 1: **WF11** collusion witnesses (agent pair, SoD pair, dependence)	It. 1 shared principal	Iteration 2: **WF11**	It. 2 shared principal
UC1 Meridian Insurance	[2] Fraud screening agent / Claim intake agent on Assess damage ‖ Decide claim (delegation chain); Claim intake agent / Fraud screening agent on Check fraud indicators ‖ Flag claim (delegation chain)	4	[0] none	1
UC2 Northgate Retail Bank	[1] Screening agent / Onboarding agent on Screen customer ‖ Open account (delegation chain)	2	[0] none	1
UC3 Saint-Aubin University Hospital	[0] none	2	[0] none	1
UC4 Regional Benefits Agency	[1] Casework assistant / Eligibility agent on Compute entitlement ‖ Decide eligibility (delegation chain)	0	[0] none	0
UC5 Helix Cloud Platform Operations	[0] none	4	[0] none	1

23.8 Compensable autonomy on the five cases

The reversibility profile was applied to each case with one set of Irreversible and Compensation facts, chosen on domain grounds and listed in Table 23.9: payment rails, gateways and ledgers are irreversible for the actor that writes them; cluster configuration and directory entries are irreversible only in the absence of the roll-back and revocation steps that the operations model already contains; an electronic prescription is irreversible once dispensed. Three organisations exhibit uncompensated autonomy. In Northgate the refund step exists but is *uncovered* in iteration 1, so the compensation is unavailable and the straight-through payments agent's autonomy is uncompensated; iteration 2 covers the step and the defect disappears without touching the payments charter. In Meridian and Saint-Aubin no reversal step exists in the model at all: the settlement agent's autonomy over the payment gateway and the ledger, and the coding-and-billing agent's over the billing system, remain uncompensated in iteration 2, and the finding is that a step is missing from the process model, not that a charter is wrong. Helix and the Benefits Agency are compensated in both iterations because roll-back, revocation and returns-reconciliation steps are covered. The table is generated by `wf12_arp_demo.py`.

Table 23.9. **WF12** compensable autonomy on the five cases: reversibility facts declared, Uncompensated witnesses (agent on resource) per iteration, and the number of compensated autonomous authorities; computed by `wf12_arp_demo.py`.

Case	Irreversible	Compensation ($x \to s_c$)	Iteration 1: Uncompensated [count] agent on resource (step)	Iteration 2
UC1 Meridian Insurance	Payment gateway, General ledger	none modelled	[2] Settlement agent on Payment gateway (Settle payment); Settlement agent on General ledger (Settle payment) (compensated: 0)	[2] Settlement agent on Payment gateway (Settle payment); Settlement agent on General ledger (Settle payment) (compensated: 0)
UC2 Northgate Retail Bank	Payment rail	Payment rail → Refund customer	[2] Payments agent on Payment rail (Initiate transfer); Payments agent on Payment rail (Release payment) (compensated: 0)	[0] none (compensated: 1)
UC3 Saint-Aubin University Hospital	Billing system, e-prescribing system	none modelled	[2] Coding and billing agent on Billing system (Submit claim); Coding and billing agent on Billing system (Post payment) (compensated: 0)	[1] Coding and billing agent on Billing system (Post payment) (compensated: 0)
UC4 Regional Benefits Agency	Banking gateway	Banking gateway → Reconcile returns	[0] none (compensated: 1)	[0] none (compensated: 1)
UC5 Helix Cloud Platform Operations	IAM directory, Cluster configuration, Cloud billing account	IAM directory → Revoke expired access; Cluster configuration → Roll back	[0] none (compensated: 5)	[0] none (compensated: 2)

23.9 The Authority Request Protocol on the five cases

The protocol was exercised on the iteration-1 models by generating the requests an agent population would submit: one *well-founded* request for every uncovered agentic step (the nearest chartered agent asks for the step at its required level) and two kinds of *ill-founded* request (an autonomous agent asks for autonomous authority on a reserved step of its process; an agent asks for the other side of a separation-of-duty pair it already covers). Each request was pre-analysed as a Proposed charter (step 3 of Chapter 16). Table 23.10 gives the outcome: all seven well-founded requests are forwarded to their principal with a positive coverage delta and no new defect; all eleven ill-founded requests are returned automatically, seven by **WF8**, three by **WF4** and two of those additionally by **WF11** (the requesting agent would have come to control both sides of a regulated pair through its delegation). No human was consulted for a request the model could refuse, and no request the model could not refuse was decided by the model.

Table 23.10. Authority Request Protocol on the five iteration-1 models: requests generated, pre-analysis decision and the condition that returns each ill-founded request; computed by `wf12_arp_demo.py`.

Case	Well-founded requests (uncovered agentic step, nearest chartered agent, required level): submitted / forwarded to principal	Ill-founded requests (autonomous on a reserved step; other side of an SoD pair): submitted / returned by pre-analysis, with the condition that returns them
UC1 Meridian Insurance	1 / 1 forwarded: Customer reply agent → Send reply @commit (+1 cov.)	2 / 2 returned: Settlement agent → Decide claim @autonomous: WF8; Claim intake agent → Flag claim @commit: WF4/WF11
UC2 Northgate Retail Bank	2 / 2 forwarded: Dispute agent → Register dispute @commit (+2 cov.); Dispute agent → Refund customer @commit (+2 cov.)	3 / 3 returned: Payments agent → Decide SAR @autonomous: WF8; Alert triage agent → File SAR @commit: WF4; Screening agent → Open account @commit: WF4/WF11
UC3 Saint-Aubin University Hospital	1 / 1 forwarded: Discharge-draft agent → Send discharge letter @commit (+2 cov.)	1 / 1 returned: Coding and billing agent → Sign note @autonomous: WF8
UC4 Regional Benefits Agency	2 / 2 forwarded: Appeals agent → Register appeal @commit (+1 cov.); Appeals agent → Implement ruling @commit (+1 cov.)	2 / 2 returned: Payment agent → Approve payment run @autonomous: WF8; Fraud screening agent → Approve payment run @autonomous: WF8
UC5 Helix Cloud Platform Operations	1 / 1 forwarded: Capacity agent → Scale cluster @commit (+1 cov.)	3 / 3 returned: Remediation agent → Approve change @autonomous: WF8; Provisioning agent → Approve change @autonomous: WF8; Monitoring agent → Apply remediation @commit: WF4

23.10 The v1.1 profiles on the five cases

New in Charterion v1.1

The five case models are unchanged. For each, the author proposed the facts of the seven v1.1 profiles—reviewed and running configuration digests, instances and cardinalities, performer roles for reserved steps, delegation caps, obligations with instrument labels, collaboration edges and a depth bound, blast bounds per agent class—together with a synthetic decision log of two records, per-slice execution levels and per-relationship exposure levels for **CC4**, and validity on delegations for **WF10**; they are declared in `cases/uc_v11_facts.py` and labelled *author-proposed, for demonstration*. Both iterations were evaluated at the instant 15 November 2026, inside the validity windows of the refined charters; the v1.0 tables of this part are not re-evaluated at an instant and are unchanged. Every number below is generated by `run_cases_v11.py` from `cases_results_v11.json`, and the runner re-parses each table cell and asserts it against the JSON before writing, as in v1.0. The evidence kind is *demonstrative*.

Reading. Iteration 1 shows, in every case, one configuration drift (an agent redeployed under a new model after its charter was reviewed), and in three of five an out-of-perimeter delegation and a cap amplification; in every case at least one agent exceeds the blast bound of its class, and one or two agents per case still exceed it in iteration 2 because their step-scoped charters still reach the resources their processes require—the bounds proposed are deliberately tight, and the resolution would be a bound review, not a re-scoping. Unstaffed reserved steps appear where the v1.0 models already had vacant roles (UC3, UC4, UC5) and survive refinement, since refining charters does not fill a vacancy. The unmet obligations of iteration 1 are the autonomous charters of the broad first pass falling inside the scope of a human-decision obligation; refinement to steps removes them in UC1 and UC5 and leaves them where an autonomous charter remains on a step in scope (payment initiation in UC2, payment posting in UC3, payment suspension and the payment run in UC4). UC5 iteration 2 shows the *Uncharted* pattern: the cron bot's charter was retired in the refined model while an instance of it is still running. The synthetic decision logs give one attributed and one unattributed record per case, the latter an agent acting on a resource that no charter of its explains.

Regulatory evidence. With the obligation profile in use, the Regulatory Evidence Report of Appendix M was generated for each model: 22 obligations per iteration across the five cases (human-decision, record and oversight kinds; instrument labels are the author's), with the evidence computed from the model—steps in scope, reservation and staffing, agents and levels, decision-log attribution, overseers and reachability—and a per-obligation verdict of whether the model satisfies it (`cases/regulatory_evidence_cases.md`). The legal reading of the verdict is not the framework's.

Table 23.11. The v1.1 conditions on the five cases: defects per condition, iteration 1 (broad charters) and iteration 2 (refined), evaluated at 15 November 2026 with all v1.1 profiles in use; also the three v1.0 conditions completed in the v1.1 toolkit (**WF10** on delegations, **CC4**, **CC6**)

Case	It.	v1.1 facts	Config.	Inst.	Staff.	Caps	Oblig.	Perim.	Blast	WF10(d)	CC4	CC6	CC7
UC1 Meridian Insurance	1	39	2	1	0	1	1	1	2	1	1	1	5
UC1 Meridian Insurance	2	39	3	0	0	0	0	0	2	1	0	1	5
UC2 Northgate Retail Bank	1	43	2	1	0	0	6	1	2	0	1	0	7
UC2 Northgate Retail Bank	2	54	0	1	0	0	1	0	2	0	0	0	2
UC3 Saint-Aubin University Hospital	1	36	1	1	2	0	4	0	1	1	1	1	5
UC3 Saint-Aubin University Hospital	2	38	0	1	2	0	1	0	1	0	0	1	2
UC4 Regional Benefits Agency	1	38	1	1	1	1	5	2	1	0	1	1	18
UC4 Regional Benefits Agency	2	40	2	1	1	0	2	0	1	0	0	1	2
UC5 Helix Cloud Platform Operations	1	43	2	1	1	1	1	0	3	1	1	0	6
UC5 Helix Cloud Platform Operations	2	48	0	2	1	0	0	0	1	0	0	1	2

Reproducing. `run_cases_v11.py` regenerates `cases_results_v11.json` and Tables 23.11–23.14; `regulatory_evidence.py` the evidence reports; `render_wf12_table_v11.py` re-renders Table 23.9 with the step of each witness. The v1.0 case files, runner and demonstration script are shipped unchanged.

23.11 Reproducing the cases

The case models (`cases/bank.py`, `hospital.py`, `benefits.py`, `itops.py` and the Meridian reference `casestudy.py`) and the runner `run_cases.py` are supplied with the toolkit. Running the runner regenerates `cases_results.json`, every table of this part and the case narratives, and `charterion_separation.py` computes Table 23.8 and `wf12_arp_demo.py` Tables 23.9 and 23.10; the runner re-parses each generated table cell and asserts it against the JSON before writing. The Meridian extension facts used for **WF8** (reserved steps: the claim decision and the SIU referral; the ledger posting left undecided) are a v1.0 addition to the published case and are declared as a constant in the runner.

Table 23.12. Iteration-2 witnesses of the v1.1 conditions, in business terms (counts in brackets); **CC7** *InvalidCertificate* witnesses are the envelope entries of agents without an accountable human (**WF1**), for which no certificate can be valid, plus one entry per case deliberately certified against the other iteration's snapshot

Case	Condition	Iteration-2 witnesses (counts in brackets)
UC1	WF13 ConfigDrift	[3] Fraud screening agent (charter m3a); Fraud screening agent (charter m3b); Settlement agent (charter m6)
	WF19 BlastExceeded	[2] Settlement agent; Legacy bot
	WF10 Stale(d)	[1] Reconciliation agent → Ledger agent on Monthly close
	CC6 UnattributedEvidence	[1] R2
	CC7 Uncertified	[1] Uncertified_demo
	CC7 InvalidCertificate	[4] Claim intake agent on Claims DB (commit); Legacy bot on Customer CRM (read); Legacy bot on Policy DB (commit); Legacy bot on Email service (commit)
UC2	WF14 OverInstantiated	[1] Screening agent
	WF17 UnmetObligation	[1] o3: Payments agent autonomous on Initiate transfer
	WF19 BlastExceeded	[2] Onboarding agent; Screening agent
	CC7 Uncertified	[1] Uncertified_demo
	CC7 InvalidCertificate	[1] Payments agent on Customer master (read)
UC3	WF14 OverInstantiated	[1] Patient communications agent
	WF15 UnstaffedReserved	[2] Prescribe; Validate triage
	WF17 UnmetObligation	[1] o3: Coding and billing agent autonomous on Post payment
	WF19 BlastExceeded	[1] Coding and billing agent
	CC6 UnattributedEvidence	[1] R2
	CC7 Uncertified	[1] Uncertified_demo
	CC7 InvalidCertificate	[1] Scheduling agent on Electronic health record (propose)
UC4	WF13 ConfigDrift	[2] Fraud screening agent (charter m7b); Fraud screening agent (charter m7c)
	WF14 OverInstantiated	[1] Fraud screening agent
	WF15 UnstaffedReserved	[1] Decide eligibility
	WF17 UnmetObligation	[2] o2: Fraud screening agent autonomous on Suspend payment; o3: Payment agent autonomous on Execute payment run
	WF19 BlastExceeded	[1] Payment agent
	CC6 UnattributedEvidence	[1] R2
	CC7 Uncertified	[1] Uncertified_demo
	CC7 InvalidCertificate	[1] Intake agent on Case management system (commit)
UC5	WF14 Uncharted	[1] instance i_cron_1
	WF14 OverInstantiated	[1] Remediation agent
	WF15 UnstaffedReserved	[1] Approve capacity purchase
	WF19 BlastExceeded	[1] Change agent
	CC6 UnattributedEvidence	[1] R2
	CC7 Uncertified	[1] Uncertified_demo
	CC7 InvalidCertificate	[1] Monitoring agent on Telemetry (read)

Table 23.13. Authority certificates on the five cases: certificates issued for every NEED tuple of the charter-derived envelope, checked against the same snapshot; rejections split by cause; one uncertified approved entry per case demonstrates **CC7**

Case	It.	Certificates issued (= Need tuples)	Valid	Rejected: stale model digest	Rejected: no accountable human (WF1)	Uncertified entries (CC7)
UC1 Meridian Insurance	1	36	32	1	3	1
UC1 Meridian Insurance	2	25	21	1	3	1
UC2 Northgate Retail Bank	1	31	25	1	5	1
UC2 Northgate Retail Bank	2	23	22	1	0	1
UC3 Saint-Aubin University Hospital	1	25	21	1	3	1
UC3 Saint-Aubin University Hospital	2	21	20	1	0	1
UC4 Regional Benefits Agency	1	33	16	1	16	1
UC4 Regional Benefits Agency	2	27	26	1	0	1
UC5 Helix Cloud Platform Operations	1	30	25	1	4	1
UC5 Helix Cloud Platform Operations	2	20	19	1	0	1

Table 23.14. Agency debt on the five cases (Table 23.7-bis, supersedes Table 23.7): the seven v1.0 components including $\mathrm{AD}_{\mathrm{stale}}$ (zero at the evaluation instant) and the v1.0 index unchanged; the six informative v1.1 components (drift, unstaffed, amplified, unmet obligation, perimeter, blast) and the informative index $\mathrm{ADI}_{1.1}$ reported beside it, per agentic step, unweighted

Case	It.	Lat.	Res.	Shad.	Cont.	Orph.	Unw.	Stale	ADI	Drift	Unst.	Ampl.	Oblig.	Perim.	Blast	$\mathrm{ADI}_{1.1}$
UC1 Meridian Insurance	1	3	3	2	8	1	1	0	1.385	1	0	1	1	1	2	0.462
UC1 Meridian Insurance	2	0	0	2	4	1	1	0	0.615	3	0	0	0	0	2	0.385
UC2 Northgate Retail Bank	1	4	4	1	14	1	1	0	1.190	1	0	0	6	1	2	0.476
UC2 Northgate Retail Bank	2	0	0	1	1	0	0	0	0.095	0	0	0	1	0	2	0.143
UC3 Saint-Aubin University Hospital	1	5	3	2	9	1	1	0	1.235	1	2	0	4	0	1	0.471
UC3 Saint-Aubin University Hospital	2	0	0	0	0	0	0	0	0.000	0	2	0	1	0	1	0.235
UC4 Regional Benefits Agency	1	4	4	2	14	4	1	0	1.812	1	1	1	5	2	1	0.688
UC4 Regional Benefits Agency	2	0	0	0	1	0	0	0	0.062	2	1	0	2	0	1	0.375
UC5 Helix Cloud Platform Operations	1	6	6	2	15	1	1	0	1.632	1	1	1	1	0	3	0.368
UC5 Helix Cloud Platform Operations	2	0	0	0	0	0	0	0	0.000	0	1	0	0	0	1	0.105

Part VIII

Positioning

24 Positioning against existing frameworks

24.1 The claim

Much of what an agent needs has been articulated, since 2025, by security and standards bodies: that an agent is an identity class of its own with a human sponsor; that its rights should be scoped, temporary and least-privileged; that delegation between agents must not amplify privilege and must trace to a human; that autonomy comes in levels with controls attached; that actions must be intercepted, logged and reversible at run time. Charterion adopts every one of these positions and claims none of them as its own. Its claim is narrower and, the author believes, more consequential for the discipline of enterprise architecture:

24.2 The five central contributions

Charterion is not to be defended as a list of two dozen novelties. Five contributions carry the framework; every other construct is a consequence of one of them and is presented as such.

1. **Enterprise architecture as an authority substrate.** The enterprise model is written for a program that acts and cannot interpolate; it is the origin of every authority an agent holds and the addressee of every admissibility question (Chapter 3).
2. **Business-scoped authority formally derived into technical entitlement.** An agent's entitlement on a system is a theorem of its charters on capabilities, processes and steps, not a declaration in an identity store; narrowing a charter narrows the entitlement (**WF3**, D5, **CC1**–**CC2**).
3. **A formal semantics for agent authority inside the enterprise metamodel.** Charter, delegation, accountability, coordination and coherence have a stratified Datalog semantics with a least model, proofs and a reference reasoner that conforming tools must agree with (Chapter 7, Appendix A).
4. **Design–run-time–boundary coherence.** Six conditions make "the run time enforces what the architecture justified" a property of data checked on every change, at home and across organisations (Chapter 11, Chapter 10).
5. **Machine-addressed architecture with explicit epistemic warrant.** Every assertion carries provenance, confidence, validity and status; the architecture can answer *unwarranted* instead of guessing, and every decision is traceable to the assertions that warranted it (Chapter 9).

New in Charterion v1.1

Version 1.1 and the five contributions. v1.1 sharpens each contribution without adding a sixth. (1) The substrate premise is now qualified against the machine-addressed EA repositories of three vendors (Section 24.5): what makes a repository a substrate is that authority originates in it and admissibility is asked of it, neither of which any of the three does. (2) Business-scoped derivation gains an independently checkable object per entitlement (Section 11.5). (3) The formal semantics gains a stratum and a general preservation theorem (Proposition 8.4), and its extension mechanism is exercised seven times. (4) Design-time coordination gains the perimeter and staffing profiles and the machine-verified regression that the v1.0 conditions are unchanged. (5) The warrant claim is narrowed to the four elements no retrieved work matches. The five contributions remain the whole of the claim.

The reserved-step, reachability, temporal, independent-separation and compensable-autonomy conditions (**WF8**–**WF12**), the Authority Request Protocol and the agency-debt metrics are secondary contributions that follow from the first premise; they are the places where the premise meets a failure specific to agentic enterprises.

Novelty claim of Charterion v1.0

Charterion proposes a shift from enterprise architecture as a human-consumed descriptive model to enterprise architecture as a formally analysable, machine-consumable *authority substrate* for autonomous agents, binding business-scoped authority, technical entitlement, run-time admissibility, human accountability and cross-organisational delegation within one coherent enterprise model (Chapter 3). Within that shift, and to the best of the author's knowledge, no existing enterprise-architecture framework or agentic-AI governance framework integrates agents as first-class principals with *business-scoped* authority, *cross-layer derivation* of resource needs from the realisation structure of the enterprise model, attenuating agent-to-agent delegation with *propagated human accountability*, and *design-time analysis* of multi-agent coordination, within a single formally analysable enterprise metamodel—and none binds that design-time model to run-time admissibility and to cross-organisational mandates by conditions a repository can verify on every change. This combination, the coherence conditions, the reserved-step, reachability, temporal, independent-separation and compensable-autonomy conditions (**WF8–WF12**), the Authority Request Protocol, the agency-debt metrics and the EA artefacts that make it adoptable, are what Charterion adds. The framework is not the first to speak of agents, charters, delegation, oversight or autonomy levels, and does not claim to be.

24.3 Classical EA frameworks

Table 24.1 compares Charterion with the classical frameworks on the ten gaps of Section 2.4. The comparison is of *published specifications*, read against the needs of an agentic-driven enterprise; each framework serves the purpose it was designed for, and Charterion relies on them for that purpose (ArchiMate for the core model, the ADM for the engagement cycle in which the design loop is embedded).

24.4 Agentic-AI governance, identity and runtime frameworks, 2025–2026

The works most likely to be raised against Charterion's claim are compared feature by feature in Table 24.2. The rows are the constructs of Charterion; the columns are the works. A filled mark means the work states the construct as a requirement or provides it; a half mark means it names the concern without a construct; a dash means the concern is out of its scope. CSA Lab Space documents are labelled by CSA as unofficial, AI-assisted research and are read here as such (Cloud Security Alliance, 2026a,b). The marks for the CSA Agent Identity Governance Framework (Cloud Security Alliance, 2026a) reflect its published scope as stated in the CSA announcement and in secondary sources—agents as a distinct identity class with a human sponsor, temporary scope-limited rights, and delegation chains that may not amplify privilege and must trace to a human principal; the primary text could not be retrieved by the author's tooling at the time of writing, so no specific requirement is attributed to it here and the marks are to be confirmed against the primary text before any external publication of this table. The term *agent charter* is also in use in Microsoft's Cloud Adoption Framework for AI agents (Microsoft, 2025), where it denotes a governance document stating an agent system's responsibilities, roles and prohibited actions; Charterion's charter is a formal tuple with a semantics, but the word is shared and the intent is the same.

Table 24.1. Coverage of the ten gaps by classical EA frameworks and Charterion (● covered, ○ partially or by convention, – not covered)

Gap	Zachman	TOGAF 10	ArchiMate 3.2	FEAF	DoDAF	Charterion
G1 Non-human principal	–	–	○ (application component; specialisation possible)	–	○ (performer)	●
G2 Authority as architecture content	–	○ (security architecture guide)	–	–	○	●
G3 Formal semantics for agent authority, delegation, coordination and run-time coherence	–	–	○ (metamodel; ontological formalisations exist in research, none for agent authority)	–	○ (DM2 ontology)	●
G4 Continuous, per-change machine-evaluated method	–	○ (iterative ADM with Phase H change management; no per-change machine-evaluated agent governance)	n/a	–	–	●
G5 Model says what it does not know	–	–	–	–	–	●
G6 Run-time decision role	–	–	–	–	–	●
G7 Human oversight as construct	–	○ (principle)	–	–	–	●
G8 SoD and coordination as architecture	–	–	–	–	–	●
G9 Inter-organisational agent authority	–	○ (Enterprise Continuum, partners)	○ (collaboration)	–	○	●
G10 Evidence return from operations	–	○ (Phase H)	–	–	–	●

24.5 The landscape re-read from primary texts, and the claim restated

New in Charterion v1.1

Table 24.2 of v1.0 was compiled from ten works read in 2026 without a written coding protocol; v1.0 itself said (Section 24.4) that the marks for the CSA Agent Identity Governance Framework had to be confirmed before external publication. For v1.1 every column was re-coded from the primary text retrieved on 6 September 2026 under the protocol of Appendix I (one mark and one verbatim excerpt of at most 25 words per cell; three marks; tie-break rules; retrieval status per work), and eleven works absent from v1.0 were added: two classical authorisation standards, three agent identity and delegation protocols, two analyst and vendor frameworks, the NIST AI Agent Standards Initiative and the MCP servers of three enterprise-architecture tools. Tables 24.2 and 24.3 replace Table 24.2. The coding was AI-assisted with a written adjudication record; the human second reading that limitation 3 of Section 26.4 asks for remains to be done, and the 27 cells of the CSA Agent Identity Governance Framework column stay provisional because its page is served only to browsers.

What changed. Of the 230 cells of v1.0, 51 differ after adjudication: 40 upward—the landscape provides more than v1.0 credited—and 11 downward on fully retrieved works, where v1.0 credited a construct the primary text does not contain (the SAIF pages, for instance, state neither an identity class for agents nor a delegation construct; the "human controllers" principle is on Google's blog). Three rows change the reading of the framework's own position and are corrected below.

Per-assertion warrant (*R15*). v1.0 marked every column "–". The *outcome* "cannot decide for lack of warranted information" exists: AARM's DEFER, the Agent Control Standard's `defer` with a `low_confidence` reason, XACML 3.0's INDETERMINATE with a `missing-attribute` status, and the trust levels of the Agent Identity Protocol on completion attestations. What remains unmatched is narrower and is now the claim: the warrant tuple $\langle \sigma, \kappa, \tau, \varepsilon \rangle$ as an *input* to authorisation, the monotonicity property P3, validity τ as a first-class element, and the separation of *unwarranted* from *escalate* and from *deny* (DEFER conflates the first two).

Agent-initiated authority request (*R20*). v1.0 marked two "○". Run-time step-up through a human exists as a mechanism in three works: the Agent Control Standard's `ask` outcome routed to a human approver and its `intent_extension`, the CIBA-based flow of the IETF AIMS draft (§10.7), and the escalation to user confirmation of South et al. What the Authority Request Protocol adds, and now claims only: the request is pre-analysed against the *design model* (**WF1**, **WF4**, **WF8**, **WF11**, **WF12**) before a human sees it; it is decided through the charter lifecycle; what results is a *charter*, not a session grant; and it is registered (Chapter 16).

Decision outcome beyond permit and deny (*R24, new row*). Four-valued admissibility is not novel and v1.0 did not claim it; Chapter 9 nevertheless presented it without positioning. XACML 3.0 has had PERMIT, DENY, INDETERMINATE and NOTAPPLICABLE since 2013; AARM has ALLOW, DENY, MODIFY, STEP UP, DEFER; the Agent Control Standard has `allow`, `deny`, `modify`, `ask`, `defer`; AuthZEN 1.0 is binary. What is specific to Charterion's function is that *escalate* is routed to a human named in the model (Section 8.2), that *unwarranted* is grounded in assertion warrants, and the outcome order of the two-sided function (Section 10.4).

Where the landscape remains empty. On the four rows that concern the agent as an element of the *enterprise*—authority anchored in the business model (R9), needs derived through it to entitlements (R10), a formal semantics with proofs (R11), design-time separation and contention analysis on a model (R12)—no work of the twenty-one provides the construct; eight cells name the concern (Microsoft's Cloud Adoption Framework asks for agent "responsibilities that map to business objectives" and "roles that prevent functional overlap"; the CSA Agent Identity Governance Framework, South et al. and the IETF AIMS draft mention the business context that justifies privileges, and AIMS states that deriving authorisation requirements from a mission is out of its scope). On design-time/run-time coherence (R17) five works name the declared-versus-observed concern (the Agent Control Standard's AG-BOM distinguishes framework-declared from runtime-discovered components; AIMS §13 audits observed behaviour against the deployment policy); none states a checkable condition between an enterprise model and run-time policy.

Enterprise-architecture repositories are machine-addressed today. Three EA-tool vendors expose their repositories to agents through MCP servers: SAP LeanIX (generally available 4 November 2025; read and write of fact sheets under the invoking user's permissions, with completeness score and quality seal exposed as data), Ardoq (generally available September 2025; reads, and writes staged in a Scenario, under a scoped API token) and Bizzdesign (announced 29 January 2026; marketing pages only). In none of the three is the repository consulted at decision time to authorise or bound an action; in none is an agent's permission derived from the architecture

Table 24.2. Constructs of Charterion against agentic-AI governance, identity and runtime frameworks of 2025–2026 (Table 24.2-bis, part A: the ten works of v1.0, marks re-coded from primary texts retrieved 6 September 2026 under the protocol of Appendix I). • provides the construct; ○ names the concern without a construct; – out of scope. † provided for a narrower object than Charterion's (qualifier in Appendix I); ? primary text read through excerpts only, mark provisional; 1.1 added in v1.1. Rows R24–R26 are new in v1.1.

Construct	Charterion	CSA AIGF?	CSA Levels 2.0	CSA ATF	AARM	OWASP ACS	OWASP T10	NIST NCCoE	IMDA 1.5	SAIF	AWS
Agent as non-human principal	•	•	•?	•?	•	•	○	•?	•	–	•
Human sponsor / accountable human	•	•	•?	•?	•	○	○?	•?	•	•	○
Scoped, least-privilege authority	•	•	•	•?	•	•	•?	•?	•	•	•
Agent-to-agent delegation, non-amplifying	•	•?	•?	○?	○	•†	○?	○	○	–	○
Autonomy / action levels	•	○?	•	•	•†	–	–	○	•	○	•
Human oversight construct	•	•?	•	•	•	•	•?	•?	•	•	•
Run-time interception / decision	•	–	•	○	•	•	○?	–	○	•†	•
Action receipts / logging	•	•	•	•	•	•	○?	○	•	•	•
Authority anchored in EA (capability, process, step)	•	–	–	–	–	–	–	○?	–	–	–
Needs derived cross-layer to entitlements	•	○	–	–	–	–	–	–	–	–	–
Formal semantics with proofs	•	–	–	–	○	–	–	–	–	–	–
Design-time SoD and contention analysis	•	–	–	–	–	–	–	–	–	–	–
Reserved human steps as model condition	•	–	○	–	○	○	–	–	•†	–	○
Temporal authority (validity, degradation)	•	•	•	○?	•†	•†	–	○	•†	–	•†
Per-assertion warrant; unwarranted outcome	•	–	–	○	•†	•†	○	○	○	○	○
Inter-organisational mandates, exposure, trust	•	○	–	○?	–	–	–	○?	○	–	○
Design-time / run-time coherence conditions	•	–	–	–	–	○	–	–	–	–	–
Independent separation across delegation chains	•	–	–	–	–	–	○	–	–	–	–
Reversibility as a model condition	•	–	•†	–	○	–	–	–	○	–	–
Agent-initiated authority request through human gates	•	○	○	–	○	•†	–	–	○	–	○
EA artefacts, viewpoints, exchange format	•	–	–	–	–	○	–	–	–	–	–
Threat model / attack taxonomy	–	○	○	•?	•	○	•	○	○	•	•
Credential and attestation mechanics	–	•	–	•?	•	•	○	•?	•	○	•
Decision outcome beyond permit/deny	•	–	–	–	•	•	–	–	–	–	–
Agent configuration inventory (AG-BOM)	•1.1	–	–	–	–	•	○	–	○	–	–
EA repository consulted by agents at decision time, with warrant / derived authority	•	–	–	–	–	–	–	–	–	–	–

Table 24.3. Table 24.2-bis, part B: eleven works absent from v1.0 — classical authorisation standards (XACML 3.0, AuthZEN 1.0), agent identity and delegation protocols (AIP, South et al., IETF AIMS), analyst and vendor frameworks (Forrester AEGIS, Microsoft CAF), the NIST AI Agent Standards Initiative, and the MCP servers of three EA-tool vendors. Marks as in part A.

Construct	XACML 3.0	AuthZEN	AIP	South et al.	IETF AIMS	AEGIS	MS CAF	NIST AASI	LeanIX	Ardoq	Bizzdesign
Agent as non-human principal	○	○	●	●	●	●	○	○	○	○	○
Human sponsor / accountable human	–	–	○	●	○	●	○	–	–	–	–
Scoped, least-privilege authority	●	–	●	●	●	●	●	–	●	●	○
Agent-to-agent delegation, non-amplifying	○	–	●†	○	●	●	○	–	–	–	–
Autonomy / action levels	–	–	–	–	–	○	–	–	–	●	–
Human oversight construct	–	–	○	●	●	●	●	–	●	●	–
Run-time interception / decision	●	●	●	●	●	●	●	–	○	○	○
Action receipts / logging	–	–	●	●	●	●	●	–	○	○	–
Authority anchored in EA (capability, process, step)	–	–	–	–	–	–	○	–	–	–	–
Needs derived cross-layer to entitlements	–	–	–	○	○	–	–	–	–	–	–
Formal semantics with proofs	–	–	○	–	–	–	–	–	–	–	–
Design-time SoD and contention analysis	–	–	–	–	–	–	○	–	–	–	–
Reserved human steps as model condition	–	–	–	○	–	○	●	–	–	○	–
Temporal authority (validity, degradation)	●	○	●	●	●	●	–	–	○	–	○
Per-assertion warrant; unwarranted outcome	●†	○	●†	○	○	○	○	–	○	○	○
Inter-organisational mandates, exposure, trust	○	○	●	●	●	○	–	–	–	–	–
Design-time / run-time coherence conditions	–	–	–	○	○	–	○	–	–	–	○
Independent separation across delegation chains	–	–	○	–	–	–	–	–	–	–	–
Reversibility as a model condition	–	–	–	○	–	–	–	–	○	●	–
Agent-initiated authority request through human gates	–	–	–	●†	●†	○	–	–	–	–	–
EA artefacts, viewpoints, exchange format	●	●	●	○	–	○	●	○	●	●	○
Threat model / attack taxonomy	●	○	●	●	○	●	○	–	–	–	–
Credential and attestation mechanics	○	○	●	●	●	●	○	○	●	●	○
Decision outcome beyond permit/deny	●	○	○	○	○	–	–	–	–	–	–
Agent configuration inventory (AG-BOM)	–	–	○	○	○	○	○	–	●	○	○
EA repository consulted by agents at decision time, with warrant / derived authority	–	–	–	–	–	–	–	–	○	○	○

model (the agent holds the invoking user's or the token's permissions); and none carries confidence or validity on its answers as a warrant. Row R26 is therefore "○" for the three and "–" elsewhere, and clause (a) of the claim is stated with this qualifier.

The novelty claim of Charterion, restated for v1.1 (informative; supersedes the claim paragraph of Section 24.2)

Charterion claims (a) the shift from an enterprise model read by people to a *machine-consumed authority substrate*—the origin of every authority an agent holds and the addressee of every admissibility question, as opposed to a repository agents may query for information, which three EA tools now offer; (b) the combination, inside one formal enterprise metamodel, of business-scoped authority, cross-layer derivation of entitlements, a stratified semantics with proved properties, and design-time separation and contention analysis, none of which the twenty-one works of Tables 24.2–24.3 provides; (c) the binding of that model to run time and to inter-organisational mandates by conditions verifiable against registers (**CC1**–**CC7**), where five works name the concern and none states a condition; and (d) in v1.1, an independently checkable derivation object for every entitlement (Section 11.5), which no work of the table carries. It does *not* claim the agent as a principal, the human sponsor, scoped or attenuating delegation, autonomy levels, human oversight, run-time interception, action receipts, run-time step-up through a human (Agent Control Standard, IETF AIMS, South et al.), agent bills of materials (Agent Control Standard AG-BOM), decision outcomes beyond permit and deny (XACML 3.0, AARM, Agent Control Standard), holder-attenuable delegation tokens (Agent Identity Protocol), or machine access to EA repositories (SAP LeanIX, Ardoq, Bizzdesign). Its per-assertion warrant claim is limited to the four elements stated above. Whether these claims survive is a matter of the evaluation protocol of Section 26.5 and of adoption; nothing in this chapter is evidence of either.

Corrections to v1.0 statements about external works. The Agent Control Standard is version 0.1.1 (specification pages 0.1.0), public since 27 May 2026 and part of the OWASP GenAI Security Project since 1 September 2026, under Apache-2.0 for specification, schemas and code and CC BY-SA 4.0 for documentation—not "released 1 September 2026". The IMDA Model AI Governance Framework for Agentic AI is at version 1.5 (20 May 2026, updated 5 June 2026). The date of 27 March 2026 that v1.0 gave for the CSA Agent Identity Governance Framework could not be confirmed (its page carries no date; it was indexed 2 April and 20 May 2026) and its "unofficial AI-assisted research" label was confirmed on sibling Lab Space documents, not on the page itself. The NIST concept paper's authors are Booth, Fisher, Galluzzo and Roberts; its comment period closed 2 April 2026; the sentence that it asks how agents might be identified in an enterprise architecture could be corroborated only through a secondary paraphrase and is retained with that caveat. AARM writes "STEP UP", not "STEP_UP". Retrieval date for all: 6 September 2026 (Appendix I, retrieval record).

Three observations followed from the table in v1.0 and are restated in Section 24.5 on the re-coded marks: on the constructs that concern the agent as an identity the landscape is dense and Charterion's contribution is the formalisation; on the constructs that concern the agent as an element of the enterprise no retrieved work provides a construct; on threat modelling and credential mechanics Charterion relies on the others (Section 24.7). The NIST concept paper's question about agents in an enterprise architecture is retained with the caveat of Section 24.5.

24.6 Enterprise-architecture and BPM research on agents

Three research strands neighbour Charterion. *Agentic business process management* (Calvanese et al., 2026) gives agents process frames within which they act and is the natural source of the coordination layer's intents and commitments; Charterion supplies what its authors identify as missing, the enterprise-level authority and accountability structure around the process. *Ontology-based EA analysis* (Antunes et al., 2014; Hinkelmann et al., 2016; Azevedo et al., 2015) established the formal-model view of EA on which Charterion builds and supplies the ontological grounding sketched in Section 8.8. *Infrastructure for AI agents* (Chan et al., 2024) and *TRiSM for agentic systems* (Raza et al., 2026) identify the need for identity, accountability and oversight infrastructure across agent deployments; Charterion is the enterprise-model component of that infrastructure. Work on LLM assistance *for* enterprise architecture—generating, checking or documenting models (Fill et al., 2023; Bernardi et al., 2024)—is complementary and orthogonal: it uses agents to maintain the model, which the operation loop admits under **Inv-1**–**Inv-3**.

24.7 Interfaces to what Charterion relies on

Table 24.4. What Charterion provides to and requires from neighbouring frameworks and systems

Neighbour	Charterion provides	Charterion requires
Identity governance (CSA AIGF, NIST NCCoE, Entra Agent ID, DIF/KYA)	The target entitlement set per agent (derived Need); the accountable humans (Acc); the agent inventory bound to charters	Stable agent identities bound to Agent constants; the provisioned state as Grant facts; credential and attestation mechanics
Autonomy-level frameworks (CSA Levels, IMDA)	The level of each charter and the conditions under which it may change (evidence-gated promotion); the mapping of Table 8.2	Control catalogues per level, which Charterion does not define
Runtime enforcement (AARM, OWASP ACS, policy engines)	The charter-derived envelope as declarative policy with charter provenance; escalation routing (Esc); caps as thresholds	Interception at the tool boundary; the decision log as $\mathcal{L}$ with supporting assertions; reversal records
Threat and risk frameworks (OWASP Agentic Top 10, SAIF 2.0, MAESTRO, AWS scoping matrix)	Blast radius and reach per agent; the delegation graph; the list of autonomous agents and their overseers as the population to threat-model	Threat taxonomies and controls; the classification of the deployment (scope) that informs the target conformance level
Agentic BPM and process engines	Agentic and reserved designations; the authority of each agent on each step	Process frames, intents and commitments for the coordination layer; control-flow order (Before)
Regulation and AI management systems (AI Act, ISO/IEC 42001, NIST AI RMF)	Evidence: oversight structure, reserved decisions, logs, evidence records, conformance statement (Chapter 19)	Classification of systems and the obligations that follow, which determine the required level per slice

Part IX

Maturity and conformance

25 Maturity model and Conformance Statement

25.1 Three scales

Charterion conformance is stated per slice on three scales, one per plane. The execution-plane scale (Levels 0–4, Section 9.6) and the exposure scale (X0–X4, Section 10.3) are inherited from the articles. The design-plane scale is new in v1.0.

New in Charterion v1.0

Design-plane conformance levels.

D0 Unmodelled
: Agents are not in the enterprise model; authority is administered in identity management and agent catalogues.

D1 Chartered
: Every agent in the slice has AGENT and CHARTER facts with principal; **WF1** and **WF2** hold or are accepted. The three architect questions can be asked.

D2 Well formed
: The core model carries REQUIRES for every agentic step; **WF1**–**WF9** and **WF11** hold or are accepted; RESERVED steps are declared; the Guard Register is complete. Grants are compared with needs (**WF3**) at least on every release.

D3 Governed
: The temporal profile is in use and **WF10** holds; **WF12** holds for every slice with an *autonomous* charter; the Authority Request Protocol is the only path by which an agent obtains authority it was not chartered for; the charter lifecycle is exercised with its gates; the design loop runs as a repository check on every change; the Accepted-Defect Register and Charter Rationale Records are complete; provisioning is performed from the model (D5).

D4 Coherent
: **CC1**–**CC6** hold against the execution plane (and the boundary plane where in use); bridge B2 returns evidence and the promotion rule is applied; agency debt is trended and bounded by a published target.

25.2 The maturity model

Table 25.1 combines the three scales into five stages that describe an organisation's overall position. The stages are descriptive; conformance is always claimed on the scales, per slice, and **CC4** constrains their combination: write-level authority requires execution Level 3, and exposure X2/X3/X4 require execution Levels 2/3/4 on the exposed slice.

25.3 Roadmap

The stages are also the recommended order of adoption for a slice, because each removes the failure the next would otherwise inherit. Stage 1 removes misidentification and orphan authority; it needs an existing repository with identifiers, the agent inventory and a principal for each agent. Stage 2 removes shadow entitlements, uncovered steps, separation-of-duty violations and uncoordinated writers; it needs REQUIRES facts, which is the substantive modelling effort, and the API catalogue joined to the anchor. Stage 3 is the governance step: the organisation's architecture board must be willing to state as rules what it states as principles, to accept that a rule it approves will refuse an agent, and to let the pipeline provision. Stage 4 is the learning step, and depends on decision logs of sufficient volume. An organisation SHOULD bring its first slice to Stage 3 before conferring *autonomous* authority anywhere, and SHALL not confer write-level authority on a slice below execution Level 3 (**CC4**).

Table 25.1. The Charterion maturity model

Stage	Design	Execution	Exposure	Characterisation
0 Documented	D0	0	X0	EA as documentation; agents governed outside the architecture, if at all. Where most practices are today (El Hassani, 2026b).
1 Inventoried	D1	1	X0–X1	Agents chartered and accountable; the anchor layer queryable; read-only agents grounded on identity.
2 Analysed	D2	2	X1–X2	Well-formed agency layer checked on every release; reversible actions on known affordances; catalogue exposed and reservations accepted from partners.
3 Governed	D3	3	X2–X3	Approved envelope derived from charters; four-valued admissibility in force; irreversible actions under model-bounded oversight; lifecycle gates exercised; binding cross-boundary commitments.
4 Coherent	D4	4	X3–X4	Design, execution and boundary planes provably coherent; agents maintain the substrate under the invariants; evidence returns and promotes; trust re-estimated from commitments; settlement with partners.

25.4 The Conformance Statement

Contents of a Conformance Statement

An organisation claiming Charterion conformance SHALL publish, per slice, a statement containing:

1. the slice (capabilities, processes, resources, agents) and the version of Charterion claimed;
2. the conformance target(s) claimed (model, method, tool) and the level on each scale;
3. the profiles in use (temporal, quantitative, reserved, escalation, reversibility, boundary; in v1.1 also configuration, instance, staffing, cap-composition, obligation, perimeter, containment, certificate) and the parameters δ and the class blast bounds where those profiles are in use;
4. the parameters: θ_{id}, θ_{act}, β, the promotion thresholds, the agency-debt weights w_i, the lifecycle cadences (review, drift detection, trust re-estimation), the decision-log retention;
5. the Accepted-Defect Register for the slice;
6. the recommendations (SHOULD) not followed and the justification for each;
7. the tool(s) used and their tool-conformance evidence;
8. the date of the last analysis and the agency-debt vector at that date.

The statement is generated from the model and the registers except for items 6 and the rationales of item 5, which are written by the roles that decided them. It is the document an auditor, a regulator or a counterparty reads first, and **CC4** is evaluated from it.

New in Charterion v1.1

Profiles recommended at each design level (v1.1; SHOULD, never SHALL). The four design-plane levels are unchanged and no v1.1 condition is required at any level (Appendix H). Table 25.2 states where each profile is recommended; an organisation that does not follow a recommendation records it under item 6 of the Conformance Statement.

Table 25.2. Recommended (SHOULD) adoption of the v1.1 profiles by design level

Profile	Level	Recommendation and reason
staffing	D2	every reserved step declared at D2 SHOULD have a performer role: reservation without a performer is coverage without a cover
perimeter	D2	the delegation graph SHOULD be declared when delegations are analysed for **WF6** and **WF11**
configuration	D3	reviewed digests SHOULD be recorded from G2 onward once the temporal profile makes charters reviewable
obligation	D3	reserved steps SHOULD be grounded where a regulatory owner exists (Chapter 19)
cap-composition	D3	with the quantitative profile
instance	D4	instance facts come from the runtime, which D4 presupposes
containment	D4	blast bounds are meaningful once **CC1**–**CC2** tie the model to provisioned grants
certificate	D4	certificates are emitted by the provisioning pipeline of D5 and checked at O2

Part X

Validation status

26 Validation, testable propositions and limitations

26.1 What has been validated, and how

Charterion is a design artefact evaluated, so far, on the artificial and formative side of the FEDS trajectory (Venable et al., 2016). Table 26.1 states, construct by construct, the kind of evidence that exists. Three kinds are distinguished: *analytical* (a proof), *experimental* (a controlled study on generated models or a simulation) and *demonstrative* (a worked case by the author). No construct has *naturalistic* evidence—use on an organisation's own repository by people other than the author—and the document says so wherever a claim might otherwise be read as stronger.

26.2 Summary of the experimental evidence

Scalability (DR6). On synthetic well-formed models from 427 to 489 952 facts, the reference reasoner's median analysis time grew from under a millisecond to 1.64 s (maximum 2.01 s); between 14 474 and 489 952 facts time grew as size to the power 1.40, the super-linear part being the pairwise contention term, and the delegation fixpoint converged in at most ten iterations at every size (El Hassani, 2026e). The analysis is therefore usable as a check on every commit for repositories far larger than any enterprise model in practice.

Projection (DR5, gaps G2 and G8). With one defect of each kind injected into each of thirty well-formed models of about 2 500 facts (240 injections), the reasoner reported the intended witness every time. The best detectors writable on the entitlement view of identity management and on the agent-catalogue view could not express **WF2**, **WF3** or **WF4** at all; the entitlement view detected contention with precision 0.79 and autonomy not at all, the catalogue view could not express contention, and both missed over-delegation in six of thirty cases (El Hassani, 2026e). This is the evidence that the cross-layer model is necessary, not merely convenient.

Granularity. Re-scoping the charters of twenty models of sixty processes: capability charters yielded 2 741 authority tuples, 611 write authorities on human steps and 2 719 contention pairs with no uncovered step; step charters 154, 17 and 27 with 84 uncovered agentic steps; process charters lay between (El Hassani, 2026e). The granularity rule of Section 7.5 follows.

Warrant (TP2). In a stylised simulation of an estate whose assertions decay at heterogeneous rates, per-assertion validity re-estimated from drift detection reduced the stale-action hazard at equal human escalation by 31 % on average (range 6–53 % across seeds) relative to a uniform validity window in the base condition, rising to 51 % (range 14–84 %) when change rates were more heterogeneous, and dominated it at every escalation level; re-observation by agents of what they act upon removed 61 % of the baseline hazard but did not suffice on its own (El Hassani, 2026c). Only the ordering of policies should be carried away from the simulation, not its absolute rates.

26.3 Testable propositions

As a nascent design theory (Gregor and Hevner, 2013), Charterion yields propositions that naturalistic evaluation can test. The propositions of the articles are retained (TP1–TP4 of the third article; XP1–XP3 of the fourth; the four propositions of the fifth) and nine are added for the v1.0 constructs.

Table 26.1. Validation status of Charterion constructs by kind of evidence

Construct	Analytical	Experimental	Demonstrative	Notes
Core semantics, Proposition 7.1–7.4	•	•	•	Proofs; scalability study to 489 952 facts (1.64 s median) (El Hassani, 2026e)
WF1–**WF7**	•	•	•	Projection study: 240 injections, all detected by the reasoner; Meridian and four new cases (Part VII)
Granularity rule	–	•	•	20 models × 3 scopes (El Hassani, 2026e); confirmed in the five cases
WF8–**WF10**, temporal profile	•	–	•	Propositions 8.1, 8.2, Corollary 8.1; toolkit tests; cases. No controlled study yet
WF11 independent separation	•	–	•	Proposition 8.3; found in 3 of 5 first-pass cases, removed in all by iteration 2 (Table 23.8)
WF12 compensable autonomy	•	–	•	Reversibility profile applied to the five cases (Table 23.9): uncompensated autonomy in 3 of 5, in 2 of which no reversal step exists in the model
Authority Request Protocol	•	–	•	Proposition 16.1; 18 generated requests on the five iteration-1 models (Table 23.10): 7/7 well-founded forwarded, 11/11 ill-founded returned. No request has yet been processed by an organisation
Admissibility function, P1–P3	•	• (TP2)	•	Simulation: per-assertion validity reduces stale-action hazard by 31 % (range 6–53 %) at equal escalation in the base condition (El Hassani, 2026c)
Lifecycle, **Inv-1**–**Inv-5**, Levels 0–4	–	–	•	Scenario of 45 assertions (El Hassani, 2026c)
Boundary plane, X-P1–X-P4, X0–X4	•	–	•	Procurement scenario (El Hassani, 2026d)
Coherence **CC1**–**CC6**, Theorems 11.1–11.2	•	–	•	Proofs; toolkit tests (v1.0 toolkit: **CC1**–**CC3**, **CC5**; **CC4**, **CC6** analytical only until v1.1—E8); not yet exercised against a live substrate
Charter lifecycle, promotion rule, agency debt	–	–	•	Cases report agency debt; thresholds are proposals
Method, artefacts, roles	–	–	•	Requirements traceability (Appendix D); survey-grounded requirements
Comparative positioning	–	–	–	Author's reading of specifications; a second analyst would strengthen it

Testable propositions added by v1.0

AP1 (Reserved steps)
In organisations that declare Reserved steps and enforce **WF8**, the share of write-level authority on human steps (latent authority) is lower after the first refine cycle than in organisations that rely on the granularity rule alone, and regulatory findings on human oversight are fewer.

AP2 (Reachability and span)
Escalation acceptance rates fall, and time-to-decision rises, as the overseer's span approaches β; organisations that enforce **WF9** show higher escalation acceptance at equal volume than those that do not.

AP3 (Temporal degradation)
Charters subject to the temporal profile are re-approved before their review date more often than charters without it, and the incidence of authority held by agents whose sponsoring role is vacant is lower.

AP4 (Coherence)
In organisations enforcing **CC1–CC2**, the share of run-time denials and unwarranted outcomes attributable to a mismatch between architecture and envelope falls to near zero after the first provisioning cycle, and hand-added envelope entries are detected within one analysis cycle.

AP5 (Evidence return)
Charters refined under bridge B2 converge (their denial and reversal rates fall below the promotion thresholds) in fewer review cycles than charters refined by periodic review without evidence records.

AP6 (Agency debt)
The Agency Debt Index of a slice predicts the rate of agent-related incidents in the following period better than the count of agents or of entitlements, and its components predict the kind of incident.

AP7 (independent separation)
Among multi-agent deployments with declared separation-of-duty pairs, the share of pairs whose two sides are controlled through one delegation chain is higher than the share of **WF4** violations, and higher in orchestrator-centred architectures than in peer architectures; incidents in which a regulated pair was executed end to end by one orchestrator are concentrated in the former.

AP8 (compensable autonomy)
Slices that enforce **WF12** exhibit shorter time-to-remediation of reversed autonomous actions than slices that confer the same autonomy without a modelled compensation step, because the compensation is an already-covered step rather than an improvised one.

AP9 (authority requests)
Where the Authority Request Protocol is in use, the rate of shadow grants (**WF3**) declines over time and the rate of well-founded requests per charter predicts subsequent re-scoping at D4; organisations without the protocol meet the same needs through grants that never enter the model.

26.4 Limitations

1. **No naturalistic evaluation.** The framework has not been applied to a real organisation's repository by anyone other than its author. Usability, the effort to reach each level, the willingness of architecture boards to state principles as rules, and the organisational feasibility of the roles are untested.
2. **Convenience sample.** The requirements inherit the bias of a survey recruited through one professional network after publication of the thesis; sympathisers are probably over-represented. The relative findings on which the design leans most (the ordering of the substrate gap, the conditions on agent maintenance) are less exposed to it.
3. **Author's comparison.** The comparative analyses of Chapter 24 are one analyst's reading of published specifications; a structured feature-comparison protocol with a second analyst would strengthen them, and the 2025–2026 landscape moves quickly.
4. **Cases are the author's.** The five cases exercise every construct and show the analyse–refine loop converging, but they were written by the designer and cannot show that a real estate can be described without loss.
5. **Simplified core.** The core omits ArchiMate constructs (events, interfaces, locations, motivation) and represents the technology layer solely by grants. Some organisations will need a richer core; the extension mechanism (new predicates in a new stratum) is designed for it, but each extension must re-establish stratification.
6. **Total order of levels.** A partial order of action levels would require a different attenuation operator; v1.0 does not provide one.
7. **Temporal profile is coarse.** Validity and review are dates; there is no modelling of schedules, of authority that

varies by time of day or by load, or of temporal constraints between steps beyond control-flow order.

8. **Quantities are outside the semantics.** Caps are carried to the envelope as thresholds but the design-plane analysis does not reason about amounts; **CC3**'s cap rule is a comparison, not an inference.
9. **Coherence needs registers that may not exist. CC1–CC6** presuppose an envelope register, a mandate register and a decision log in a form the validator can read; organisations whose runtime systems do not export them cannot claim D4.
10. **Trust re-estimation is unspecified.** The boundary plane requires $t_A(B)$ to be re-estimated from commitments but does not prescribe the estimator; XP2 of the fourth article is the proposition that a record-based estimator outperforms a static one.
11. **The new v1.0 conditions are the least validated. WF11**, **WF12** and the Authority Request Protocol have analytical and demonstrative evidence on the author's five cases only; their reversibility and request facts were chosen by the author. They are specified because the failures they address—orchestrated collusion, irreversible autonomy, authority obtained outside the model—are specific to agentic enterprises and have no classical counterpart; they are the first candidates for the controlled study of the agenda below.
12. **No threat model.** Charterion assumes the mechanisms it relies on—identity, interception, attestation—are sound; a compromised agent that acts outside the tool boundary is outside its reach, and **CC6** can only report the evidence afterwards.

26.5 Protocol for an independent comparative evaluation

New in Charterion v1.0

The reviews of this specification converge on one point: formal quality is no longer what is missing; independent evidence on a real repository is. This section fixes the protocol by which that evidence is to be obtained, so that the study can be run by someone other than the author and its results compared across organisations.

Design. Two arms on the same organisation and the same agentic slice, analysed by architects who are not the framework's author. *Arm A* (*baseline*): the organisation's current EA practice (TOGAF/ArchiMate or equivalent), its identity-and-access administration and its agent-governance controls (autonomy levels, sponsorship, logging), as documented. *Arm B*: the same practice with Charterion at design-plane level D2 or above—core model populated from the existing repository, charters written for the deployed agents, the analysis run, grants reconciled. Where a longitudinal design is possible, Arm B follows Arm A on the same slice after a washout; where only a cross-sectional design is possible, two comparable slices are assigned.

Metrics. Structural, measured on the models at the end of each arm: excess entitlements (grants above any justified need; **WF3**), shadow grants, separation-of-duty failures including voided separations (**WF4**, **WF11**), authority not traceable to a current human (**WF1**, **WF9**), stale or unbounded authority (**WF10**), uncompensated autonomy (**WF12**), reach and blast radius, Agency Debt Index. Operational, measured over a window of at least one quarter: agent-related incidents and their severity; architecture/run-time drift events (**CC1**–**CC2** violations or their baseline analogue); escalation volume, acceptance rate and time-to-decision; time-to-authorise a new agent need (ticket-to-grant in Arm A, request-to-provision in Arm B); time-to-remediate a reversed autonomous action; audit traceability (share of sampled agent actions for which the accountable human and the justifying business scope are recoverable within a fixed time by an auditor who did not build the model).

Hypotheses. The testable propositions AP1–AP9 (Section 26.3), of which the primary end points are excess entitlements, separation failures and audit traceability; the secondary are time-to-authorise, escalation quality and incidents. The predictive validity of the Agency Debt Index is tested separately: ADI at the start of the window against agent-related incidents during it, across slices.

Effort and instruments. Population time, analysis time and reconciliation time are recorded per slice. The instruments are the reference toolkit (which computes every structural metric from a Charterion-MXF file), the organisation's incident and access-request records, and an auditor's traceability exercise on a random sample of logged actions. The protocol, the metric definitions and the analysis scripts are to be registered before data collection.

Threats. Analyst effect (mitigated by architects independent of the author and by inter-rater agreement on charters); Hawthorne effect (the organisation knows it is being studied; mitigated by the longitudinal design's washout and by metrics computed from records rather than from self-report); and the modelling effort of Arm B being counted as benefit (mitigated by reporting effort separately). A single organisation cannot establish the

framework; three organisations in different sectors, of which at least one public, is the minimum the author proposes before any claim beyond "candidate" is made.

26.6 Research and development agenda

The agenda follows the FEDS trajectory towards naturalistic evaluation. (1) A pilot in one organisation's repository at Stage 2 on one slice, reporting the effort of D1 (populating Requires), the defects found on the first analysis and the number of refine cycles to a clean report. (2) A controlled study of **WF8**–**WF10** on generated models with injected reserved-step intrusions, vacancies and stale charters, in the design of the projection study. (3) An implementation of bridge B1 against an attribute-based policy engine and an agent runtime implementing AARM or the OWASP Agent Control Standard, and of **CC1**–**CC6** against their logs. (4) A second-analyst replication of the comparative analysis with a published protocol. (5) The UFO axiomatisation of the agency layer and its consistency check. (6) The ArchiMate specialisation profile submitted to The Open Group Architecture Forum as a proposed guide, and the method mapping as a TOGAF Series Guide. (7) Field tests of AP1–AP6 and of the propositions inherited from the articles.

26.7 Validation status of the v1.1 constructs

New in Charterion v1.1

Table 26.2 continues Table 26.1 for the constructs added in v1.1 and for the three v1.0 conditions whose toolkit support v1.1 completes. The evidence kinds are those of Table 26.1. Every experimental row rests on the controlled injection study of Section 26.8, on generated models; every demonstrative row rests on the five cases of Part VII with v1.1 facts proposed by the author (Section 23.10). No v1.1 construct has naturalistic evidence.

26.8 Controlled injection study and scalability of the complete analysis

New in Charterion v1.1

Limitation 11 of v1.0 said that **WF8**–**WF12** had no controlled study. v1.1 supplies one, on generated models, for every condition from **WF8** to **WF19** and for **CC4**, **CC6** and **CC7**; and it re-runs the scalability study of the third article with all strata and the certificate builder. Both are *experimental on synthetic models*: they establish that the reasoner detects what its rules define, at what cost; they say nothing about real estates.

Injection study. Thirty well-formed models were generated (seed 20260906; 75–361 facts, median 202; every v1.0 and v1.1 profile in use; zero defects on every condition before injection). For each of the fifteen kinds one defect was injected into each model—a reserved-step intrusion, an overloaded overseer, an expired delegation, a colluding delegation, an uncompensated autonomous write, a configuration change after review, an instance of a retired type, a vacated performer role, an over-delegated cap, an autonomous charter inside a human-decision scope, an undeclared delegation edge, a blast bound below the agent's reach, a slice level below its authority, an unattributed decision record, a tampered certificate—and the complete analysis was re-run. All 450 injections were detected with the injected witness (450/450); collateral defects, where they occur, are documented consequences of the injected facts (an undeclared delegation also changes reach; a vacated role also orphans the charters it held). The generator, the injection functions and the results are `injection_study.py` and `results/injection_study.json`.

Scalability. On generated models from 89 to 581 121 facts (2 560 capabilities, 15 360 agents, 76 800 charters, 74 240 envelope entries at the largest size), the v1.0 core analysis takes 1.53 s, the v1.0 extensions 5.01 s and the complete v1.1 analysis—all strata, all profiles, a certificate built and checked for every envelope entry—14.5 s at the largest size, growing roughly linearly with the number of facts across the range (`results/scalability_v11.json`; a single authoring workstation, Python standard library, no parallelism). The largest model exceeds the 489 952 facts of the v1.0 study.

Table 26.2. Validation status of the constructs added or completed in v1.1 (continuation of Table 26.1)

Construct	Analytical	Experimental	Demonstrative	Notes
Stratum 5, Proposition 8.4, Corollary 8.2	•	•	•	Proof; Appendix H regression: 40 model/instant pairs, identical v1.0 defect sets, authority, needs and metrics; 14 v1.0 toolkit files shipped byte-identical (Section 26.9)
WF13 configuration currency	•	•	•	Proposition 8.5; 30/30 injections detected; drift in every case at iteration 1 (Table 23.11)
WF14 chartered instantiation	•	•	•	Proposition 8.6; 30/30; UC5 iteration 2 retired-but-running instance
WF15 staffed reservation	•	•	•	Proposition 8.7; 30/30; unstaffed reserved steps in 3 of 5 cases
WF16 bounded cap composition	•	•	•	Proposition 8.8; 30/30; amplification in 3 of 5 first-pass cases
WF17 grounded obligation	•	•	•	Proposition 8.9; 30/30; regulatory evidence computed for 22 obligations per iteration (Appendix M); instrument labels author-proposed
WF18 delegation perimeter	•	•	•	Proposition 8.10; 30/30; out-of-perimeter delegations in 3 of 5 first-pass cases
WF19 bounded blast radius	•	•	•	Proposition 8.11; 30/30; class bounds exceeded in every case, before and after refinement (the author-proposed bounds are deliberately tight)
Authority certificates, **CC7**, Propositions 11.2–11.4	•	•	•	Soundness and completeness proofs; 30/30 tampered certificates rejected; every Need tuple of the ten case models certified and checked (Table 23.13); no audit has yet used one
WF10 on delegations, **CC4**, **CC6** (toolkit completion)	•	•	•	Specified in v1.0, implemented in v1.1; 30/30 each; **CC6** on synthetic decision logs only
Trust estimator (Appendix O)	•	–	–	Informative; monotonicity checked executably; no data
Table 24.2-bis coding protocol (Appendix I)	–	–	•	AI-assisted first reading with adjudication record; human second reading pending
Soufflé rendering of the programme	–	–	–	Delivered untested (no Soufflé binary in the authoring environment); mechanised proofs are agenda

26.9 Appendix H regression

For every v1.0 model available to the toolkit—the example MXF document and its test variants, the five case models in both iterations, as written and at a fixed instant, and ten generated models with all v1.1 facts removed (40 model/instant pairs)—the v1.1 reasoner with no v1.1 profile declared or in use produces defect sets, authority and need relations and metrics identical to the v1.0 reasoner, and the fourteen v1.0 toolkit files shipped with v1.1 are byte-identical to the v1.0 toolkit (`regression_v11.py`, `regression_report.md`). This is the mechanical form of the guarantee of Appendix H.

26.10 Testable propositions added in v1.1

Nine propositions are added for the v1.1 constructs, in the form of AP1–AP9: each names a measurable and what would falsify it. None has been tested.

AP10 (configuration drift is common)
In slices that adopt the configuration profile, at least one write-level charter shows ConfigDrift within 90 days of approval in a majority of slices; the rate falls once the pipeline records Reviewed automatically. Falsified if drift is rare ($< 10\,\%$ of charters per quarter) in slices with actively developed agents.

AP11 (retired on paper, still running)
On first import of instance facts at D2 or above, Uncharted is non-empty in slices that retired a charter in the preceding quarter. Falsified if runtimes stop instances on retirement reliably across three or more organisations.

AP12 (vacancy effects co-occur)
UnstaffedReserved and **WF1** Orphan are positively associated across slices and appear together after reorganisations; ReservedLoad concentrates on the humans with the largest oversight span. Falsified if the two are independent or the load is evenly spread.

AP13 (first-pass cap amplification)
First-pass models with the quantitative profile show CapAmplification for at least one delegating agent, removed by refinement without loss of coverage. Falsified if composed caps are declared correctly on first pass in a majority of slices.

AP14 (grounding rises with level)
The share of reserved steps grounded in an obligation rises with design level, and ungrounded reserved steps are re-designated agentic more often in the next iteration than grounded ones. Falsified if grounding is unrelated to level or to later re-designation.

AP15 (perimeters carry information)
Out-of-perimeter delegations appear in slices whose orchestrators acquire tools at run time, and refined perimeters satisfy $\delta \leq 2$ in a majority of slices. Falsified if declared perimeters are routinely as wide as the agent set.

AP16 (blast radius is skewed and fixable)
$|\text{Blast}(a)|$ is right-skewed in first-pass models; class bounds are exceeded by the top decile; step-scoped refinement brings every agent under its bound without reducing agentic coverage. Falsified if bounds can only be met by removing coverage.

AP17 (certificates shorten audits)
Per-decision audit time with certificates is constant in model size and at least an order of magnitude below re-running the reasoner; auditors accept certificate evidence for the chain of Theorem 11.1 without repository access in a majority of audited decisions. Falsified if auditors require the repository anyway or check time grows with the model.

AP18 (record-based trust beats static trust)
With the estimator of Appendix O in place of a static $t_A(B)$, the share of inbound requests that are wrongly denied or wrongly permitted at the boundary falls at equal escalation rate (the XP2 proposition of the fourth article, now with a reference estimator). Falsified if the estimator does not improve on the static value on the organisation's commitment records.

26.11 Limitations after v1.1

New in Charterion v1.1

Of the twelve limitations of Section 26.4, v1.1 closes none outright, narrows seven and leaves five as they were. Nothing in v1.1 adds naturalistic evidence.

1 No naturalistic evaluation — remains.
Unchanged; Section 26.12 now gives a field-study protocol that can be run without the author.

2 Convenience sample — remains.

3 Author's comparison — narrowed.
A written coding protocol, a per-cell evidence record and an adjudication record exist (Appendix I), and every cell was re-coded from a primary text retrieved on a stated date; the reading was AI-assisted and the human second reading with inter-rater agreement has not been done; 27 cells remain provisional.

4 Cases are the author's — remains, and the v1.1 facts are too.
The configuration, instance, staffing, cap, obligation, perimeter and containment facts of the five cases were proposed by the author for demonstration (Section 23.10).

5 Simplified core — narrowed.
Seven profiles extend the core in the manner the limitation asked for, each re-establishing stratification (Proposition 8.4); the core itself is unchanged and still omits events, interfaces, locations and motivation (obligations are the first Motivation-layer construct, Appendix E).

6 Total order of levels — remains.

7 Temporal profile is coarse — remains.
Schedules and load-dependent authority were assessed and deferred (Appendix J).

8 Quantities are outside the semantics — narrowed.
WF16 bounds the composition of permit caps along delegation at design time; amounts still do not enter the least model, and cumulative and per-period caps are deferred (Appendix J).

9 Coherence needs registers that may not exist — narrowed.
The three registers now have JSON schemas (Appendix L), the Agent Control Standard's trace and AG-BOM and the IETF AIMS audit fields standardise the run-time side, and **CC4** and **CC6** are implemented; organisations whose runtimes export none of these still cannot claim D4.

10 Trust re-estimation is unspecified — narrowed.
An informative reference estimator with the required monotonicity property is given (Appendix O); it is not normative and has no data behind it.

11 The new v1.0 conditions are the least validated — narrowed, and inherited by v1.1.
WF8–WF12 now have a controlled injection study (Section 26.8); **WF13–WF19** and **CC7** have the same kind of evidence and no more; no request has been processed by an organisation.

12 No threat model — narrowed.
The adversarial assumptions are stated (Appendix N) and **WF19** bounds the reach of a compromised agent under them; there is no adversarial evaluation, and compromise of the repository or the pipeline remains outside the framework's reach.

13 (new) Certificates prove derivation, not authenticity.
A certificate shows that an entitlement follows from recorded charters against a snapshot digest; whether the snapshot is authentic depends on a signature the specification does not define (Appendix K).

14 (new) Instrument labels are not law.
The obligation profile computes evidence against obligations the organisation records; the reading of a statute into Obligation facts is the organisation's, and the framework does not verify it.

26.12 Field-study protocol

New in Charterion v1.1

Section 26.5 fixes the two-arm comparative design. This section adds what a team needs to run it without the author: the slice selection, the instruments per phase, the data to be exported from the runtime, and the report template. It is informative.

Slice selection. One slice per organisation at Stage 2 (Section 8.9): at least three agentic processes, at least one autonomous charter, at least one delegation between agents, at least one reserved step, one runtime that exports a decision log. Exclusion: slices with fewer than eight agentic steps.

Phase and instrument. D1: the organisation's architects populate the MXF document from the repository; effort recorded per fact class; the D1 exit checklist (Chapter 13) applied. D3: the reference toolkit produces the Well-formedness Report; the architects classify every defect as *true*, *accepted* or *model error* within one week; the classification is the primary data. D4: refinement cycles counted until a clean report or acceptance. D5/O2: the envelope is exported to the identity system; the reconciliation (**CC1–CC2**) run against the identity store as it is; with the certificate profile, a sample of 30 decisions audited with certificates and 30 without, audit time recorded (AP17). Boundary arm, where applicable: mandates issued through X2 and **CC3** evaluated.

Runtime exports. Decision log (Inv-5; Appendix L schema, or the Agent Control Standard trace events mapped by Appendix K); agent configuration snapshots (AG-BOM or equivalent) for **WF13**; instance registry for **WF14**; delegation events for **WF18**.

Report. Per slice: facts by class, defects by condition and classification, cycles, effort, agency-debt vector before and after, AP1–AP18 end points, and the Conformance Statement claimed. The template is `40_annexes/field_study_report_template.md`.

26.13 Agenda after v1.1

The seven items of Section 26.6 stand; item (2) is done on synthetic models (Section 26.8) and remains to be done on real ones. Added: (8) the human second reading of Tables 24.2–24.3 with inter-rater agreement; (9) mechanised proofs of Propositions 7.1–7.4, 8.4 and 11.2–11.3 in a proof assistant, and execution of the Soufflé rendering against the Python reference on the conformance test suite; (10) a signed-snapshot binding for certificates; (11) an instance-aware core, quantitative semantics and resource partitions as v2.0 candidates (Appendix J); (12) the Open Group submission path of Appendix P.

A Consolidated formal definitions

This appendix lists, in one place and without commentary, every formal object of Charterion v1.0 with a reference to its definition.

Sorts and lattice

H, R, A, C, P, S, X, M; $\Sigma = C \cup P \cup S$; $\Pi = H \cup R$; $\mathbb{L} = \{\textit{read} < \textit{propose} < \textit{commit} < \textit{autonomous}\}$ with $\sqcap, \sqcup$; $\mathrm{pred}(\ell)$ (Definition 6.1, Definition 8.3).

Extensional predicates

Core: Human, Role, Holds, Capability, Process, Realizes, Step, Before, Agentic, Resource, Requires, Grant, Agent, Charter, Delegation, SoD, Precedence, Oversight (Table 6.1). Extensions: Reserved, Escalation, Validity, Review, Cap, Irreversible, Compensation; parameter β.

Derived predicates

Stratum 0: InScope, Ord, Conc. Stratum 1: Auth, Auth$_{\geq}$, Acc. Stratum 2: Shortfall, Covers, Need, Need$_{\geq}$, Grant$_{\geq}$. Stratum 3: Orphan, Uncovered, Shadow, Missing, SoDViol, Contention, OverDeleg, Unguarded with helpers HasAcc, Cov, Prec, AuthIn$_{\geq}$, Watched (Figure 7.2). Stratum 4 (extensions): Intrusion; Esc, HasEsc, Reachable, Span, Unreachable, Overloaded; Unbounded, Expired, Stale; Del*, Dep, Collusion, SharedPrincipal; Avail, HasComp, Uncompensated (Chapter 8). Quantities: $\mathrm{Reach}(a)$, $\mathrm{Blast}(a)$, AD, ADI (Section 8.7, Chapter 20).

Conditions

WF1–WF7 (Table 7.1); **WF8** (Condition 8.1); **WF9** (Condition 8.2); **WF10** (Condition 8.3); **WF11** (Condition 8.4); **WF12** (Condition 8.5); **CC1–CC6** (Conditions 11.1–11.6). Invariants **Inv-1–Inv-5** (Section 9.5); **X-Inv-1–X-Inv-2** (Chapter 10).

Execution and boundary objects

Assertion $\langle s, p, o, \sigma, \kappa, \tau, \varepsilon \rangle$; substrate $\langle \mathcal{A}, \mathcal{H}, \mathcal{G}, \mathcal{L} \rangle$; request $\langle g, op, x, c, t \rangle$; ADM with $\theta_{\mathrm{id}}, \theta_{\mathrm{act}}$ (Algorithm 1); $\mathrm{level}(op)$, envelope entry, charter-derived envelope (Definitions 9.4–9.5); mandate μ, exposure X_B, $t_A(B)$, κ_{eff}, commitment $\mathcal{C}$, ADM$^{\times}$ (Chapter 10).

Propositions and theorems

Least model, attenuation, accountability preservation, complexity (Propositions 7.1–7.4); intrusion locality (Proposition 8.1); temporal preservation and safe degradation (Proposition 8.2, Corollary 8.1); P1–P3 (Proposition 9.1); X-P1–X-P4 (Proposition 10.1); end-to-end accountability and boundary attenuation (Theorems 11.1, 11.2); coherence complexity (Proposition 11.1).

New in Charterion v1.1

Added in v1.1. Extensional (profiles): Config, Reviewed (configuration); Instance, Cardinality (instance); Performs (staffing); DelegationCap (cap-composition); Obligation (obligation); Collaboration, parameter δ (perimeter); BlastBound, AgentClass, ClassBlastBound (containment); validity and review on delegations (temporal). Derived, stratum 5: ConfigDrift, Unreviewed; Chartered, Uncharted, OverInstantiated, Inst, SpanI, OverloadedI; Staffed, UnstaffedReserved, ReservedLoad; OwnCap, FanOut, CapAmplification, UncappedDelegation; UnmetObligation, UngroundedReserved; OutOfPerimeter, DelegationCycle, Dist, DepthExceeded; Bound, BlastExceeded; Uncertified, InvalidCertificate (Sections 8.11–8.17, 11.5). Conditions: **WF13**–**WF19** (Conditions 8.6–8.12); **CC7** (Condition 11.7). Objects: configuration digest κ (Definition 8.8); instance (Definition 8.9); obligation (Definition 8.10); authority certificate c and $\mathrm{check}(\mathrm{EDB}, c)$ (Definition 11.1). Propositions: stratum-5 preservation and no-authority corollary (Proposition 8.4, Corollary 8.2); **WF13**–**WF19** (Propositions 8.5–8.11); certificate soundness, completeness, auditability and complexity (Propositions 11.2–11.4, Corollary 11.1). Restated: removal of collusion (Proposition 8.3, constructive form).

B Glossary

AEAF
: Working name of Charterion in the author's 2026 articles; superseded.

Accepted defect
: A failing condition the organisation has decided to live with, recorded with rationale, owner and review date.

Action level
: One of *read*, *propose*, *commit*, *autonomous*; the vocabulary of authority on all planes.

Admissibility
: The four-valued decision (permit, deny, escalate, unwarranted) of the execution plane on a request.

Agency debt
: The vector of counts of authority conferred beyond what is justified, coordinated, watched or current, and its normalised index.

Agency Grid
: The framework's summary table: three planes against the five questions of an actor.

Agency layer
: The part of the design plane that holds agents, charters, delegations and guards.

Agent
: A software system acting under conferred authority; a non-human principal.

Agentic step
: A process step the business has designated for agent performance.

Assertion
: A statement about an element or relation carrying provenance, confidence, validity and epistemic status.

Attested
: Epistemic status of content received from a counterparty under its authority.

Authority request
: A structured, pre-analysed proposal by an agent for a charter it lacks, decided by a human through the lifecycle gates (Authority Request Protocol).

Authority substrate
: The enterprise model in its Charterion role: the machine-consumable, warrant-bearing source of every authority an agent holds and every admissibility decision it receives.

Blast radius
: The resources an agent can write, including through everything it can delegate to.

Boundary plane
: The pair of substrates of two organisations and the constructs of cross-organisational agent action.

Bridge
: A transfer between loops of the method: B1 carries derived needs into the envelope; B2 carries evidence back to refinement.

Charter
: A statement that an agent may act within a business scope up to a level under an accountable principal.

Coherence condition
: A condition binding the design, execution and boundary planes (**CC1**–**CC6**).

Conformance Statement
: The per-slice declaration of what is claimed, at which level, with which parameters and exclusions.

Collusion (WF11)
: A separation-of-duty pair whose two sides are covered by agents made dependent by one delegation chain or one common delegator.

Compensation step
: The process step the organisation designates to undo or absorb an irreversible operation on a resource; required by **WF12** wherever autonomous authority is conferred on that resource.

Contention
: Two agents with write-level authority on the same resource from concurrent steps and no precedence.

Coverage
An agent covers a step if it has authority on it at or above every level the step requires.

Delegation
The passing of authority from one agent to another within a scope up to a ceiling; attenuating.

Design plane
The enterprise model with its agency layer; the source of authority.

Envelope
The layer of the substrate that states, for a principal, what it may do and where it must stop; approved content only.

Escalation human
The human who decides an agent's escalate outcomes.

Execution plane
The substrate queried at the moment of acting.

Exposure
What an organisation declares to foreign agents: an external affordance inventory and a boundary envelope; graded X0–X4.

Grant
A provisioned entitlement in identity management.

Guard
A coordination construct: separation of duty, precedence, oversight, escalation.

Latent authority
Write-level authority over steps not designated for agents.

Mandate
The portable, signed form of an agent's authority abroad.

Need A derived requirement of an agent on a resource at a level, following from the steps it covers.

Orphan
An agent with write-level authority and no accountable human.

Oversight span
The number of agents a human oversees or decides escalations for.

Principal
The human or role named by a charter as accountable.

Reserved step
A step the business reserves to human performance.

Shadow grant
A provisioned entitlement no covered step justifies.

Slice The unit of adoption and conformance: a set of capabilities or processes, their resources and agents.

Stale charter
A current charter whose review date has passed; degraded by one level.

Substrate
The set of warrant-bearing assertions, principals and decision log that agents consult.

Trust weight
The factor by which an organisation discounts a counterparty's attested confidence.

Unwarranted
The admissibility outcome meaning the substrate cannot stand behind an answer.

Warrant
The provenance, confidence, validity and status of an assertion; the fifth question of an actor.

New in Charterion v1.1

Terms added in v1.1.

Authority certificate
: An independently checkable derivation object for one entitlement: charter, delegation chain, step, scope witnesses, requirements, meet level, accountable humans, snapshot digest (Definition 11.1).

Blast bound
: The declared maximum size of an agent's blast radius, per agent or per class (**WF19**).

Cap amplification
: A delegator whose delegated permit caps, summed per unit, exceed its own (**WF16**).

Collaboration edge
: A declared permission for one agent to delegate to another; the set of edges is the delegation perimeter (**WF18**).

Configuration digest
: An opaque identifier of an agent's running configuration—model, prompt, tools, runtime (Definition 8.8); *drift* is its divergence from the reviewed digest (**WF13**).

Instance
: A running process acting under an agent type's identity, holding no charter of its own (Definition 8.9).

Obligation
: A recorded fact that an instrument imposes a human-decision, record, oversight or information obligation on a scope (Definition 8.10); *grounded* reservation is a reserved step inside such a scope.

Performer role
: The role assigned to perform a reserved step; a reserved step without a staffed performer role is *unstaffed* (**WF15**).

Profile (v1.1)
: One of configuration, instance, staffing, cap-composition, obligation, perimeter, containment, certificate; all optional, each vacuous when not declared.

Regulatory Evidence Report
: The computed, per-obligation evidence output of the obligation profile (Appendix M).

Snapshot digest
: A hash of the canonical EDB of a model, to which certificates bind.

Stratum 5
: The stratum of all v1.1 rules, below stratum 4 (Proposition 8.4).

C Notation

Symbol	Meaning
$\mathbb{L}$, $\sqcap$, $\sqcup$, pred	Action-level lattice, meet, join, predecessor
Σ, Π	Scopes ($C \cup P \cup S$); principals ($H \cup R$)
P(...)	Predicate (small capitals); $\mathrm{P}_{\geq}$ its upward closure over levels
$\leftarrow$, $\neg$, '_'	Datalog rule, stratified negation, anonymous variable
M, M_t	Design-plane model; its time-indexed form at instant t
$\langle s, p, o, \sigma, \kappa, \tau, \varepsilon \rangle$	Assertion: subject, predicate, object, provenance, confidence, validity, status
$\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{G}, \mathcal{L} \rangle$	Substrate: assertions, human principals, agent principals, decision log
$q = \langle g, op, x, c, t \rangle$	Request: agent, operation, target, context, time
θ_{id}, θ_{act}, β	Identity threshold, warrant threshold, span bound
$\mathcal{E}_t$, $\mathcal{M}_t$	Envelope entries and mandates current at t
μ, X_B, $t_A(B)$, κ_{eff}	Mandate, exposure of B, trust of A in B, effective confidence
ADM, ADM$^{\times}$, $\preceq$	Admissibility; two-sided admissibility; outcome order permit $\prec$ escalate $\prec$ unwarranted $\prec$ deny
Reach, Blast, Span, AD, ADI	Reach, blast radius, oversight span, agency-debt vector and index
WFn, **CCn**, **Inv-n**, **X-Inv-n**	Well-formedness, coherence conditions; execution and boundary invariants
DRn, **SRn**, **XRn**	Requirements of the design, execution and boundary planes
D1–D5, O1–O6, X1–X4, B1–B2	Phases of the design, operation and boundary loops; bridges

Table C.1. Notation added in v1.1 (continuation of the notation table)

Symbol	Meaning
κ, κ_r	running and reviewed configuration digest (Section 8.11); not to be confused with the warrant confidence κ of an assertion, which is always written with its tuple $\langle \sigma, \kappa, \tau, \varepsilon \rangle$
i, Inst(a), SpanI(h)	instance; instance count of a type; oversight span over instances
OwnCap(a, u), FanOut(a, u)	sum of an agent's own permit caps and of its delegated caps in unit u
δ	delegation depth bound (perimeter profile)
Dist(a, b)	shortest delegation path length
Bound(a)	effective blast bound (own or class)
c, d, digest(c), check	authority certificate, model snapshot digest, certificate self-digest, checker
ℓ^*	meet of the levels along a certificate's charter and chain
Req(s)	$\{(x, \ell) : \mathrm{Requires}(s, x, \ell)\}$
Π_5	the stratum-5 programme
$\mathrm{ADI}_{1.1}$	informative index over the six v1.1 agency-debt components
$t_A(B)$, α, β_0, h	trust estimate, Beta prior, half-life (Appendix O)

D Requirements traceability

Table D.1. Traceability of the twenty-four requirements to Charterion components and evidence

Id	Requirement (short)	Met by	Evidence
DR1	Agents as principals	AGENT; charter binds agent to scope and principal	Meridian and four cases; profile stereotype «charterion:Agent»
DR2	Business-scoped derived authority	(A1)–(A2), (N1); **WF3**; D5	Projection study: **WF3** inexpressible without the model
DR3	Traceable accountability	(K1)–(K3); **WF1**; **WF9**; **CC5**; Theorem 11.1	Proposition 7.3; cases (vacancies)
DR4	Attenuating delegation	(A2) meet; **WF6**; **CC3**; X-P2	Proposition 7.2
DR5	Design-time coordination	(O3)–(O4), (D4)–(D5); **WF4**, **WF5**	Projection study; granularity study
DR6	Tractability	Stratified Datalog; Proposition 7.4; Proposition 11.1	Scalability study to 489 952 facts
SR1	Identity	Anchor layer; θ_{id}	Scenario (El Hassani, 2026c)
SR2	Affordance	Affordance layer; $\mathrm{level}(op)$	Scenario
SR3	Authority as evaluable rules	Envelope layer; charter-derived envelope	Scenario; **CC1**–**CC2**
SR4	Coordination via shared referents	Coordination layer; commitments	Scenario; boundary scenario
SR5	Warrant on every assertion	$\langle \sigma, \kappa, \tau, \varepsilon \rangle$; **Inv-4**	Simulation (TP2)
SR6	Four-valued admissibility	Algorithm 1; P1–P3	Scenario
SR7	Instance-level currency	O1 re-observation; O6	Simulation
SR8	Machine-readable access	Charterion-MXF; validator; ADM service	Toolkit
SR9	Agent maintenance under separation	**Inv-1**–**Inv-3**; Level 4	Survey (83 %)
SR10	Federation with central commitment	O2 per-domain precedence; O3	Scenario
SR11	Non-self-authorisation	**Inv-3**; **X-Inv-1**	Survey; (Chan et al., 2024)
SR12	Accountability of decisions	**Inv-5**; **CC6**; replay	Toolkit report
XR1	Portable authority	Mandate; **CC3**	Boundary scenario
XR2	Declared exposure	Exposure; X0–X4; **CC4**	Boundary scenario
XR3	Asymmetric warrant	Attested status; $t_A(B)$; κ_{eff}	X-P3
XR4	Two-sided admissibility, local escalation	$\mathrm{ADM}^{\times}$; X-P4	Proposition 10.1
XR5	Reciprocal commitments	Commitment Ledger; X3–X4	Boundary scenario
XR6	Joint accountability	**X-Inv-1**; **CC5**; Theorem 11.2	Proof

New in Charterion v1.1

Traceability of the v1.1 conditions to the 24 requirements (no requirement added). **WF13** → **DR3**, **SR1**, **SR7** (the reviewed principal and configuration are the same; instance-level currency). **WF14** → **DR1**, **SR1**, **SR7**. **WF15** → **DR5** (design-time coordination between human and agent work), **SR9**. **WF16** → **DR4**, **XR1**. **WF17** → **DR3**, **SR12**. **WF18** → **DR4**, **DR5**. **WF19** → **DR2**, **DR6**. **CC7** → **DR3**, **SR5**, **SR12**, **XR6**. Trust estimator → **XR3** (informative). Every v1.1 construct traces to an existing requirement; Table D.1 is otherwise unchanged.

E ArchiMate 3.2 specialisation profile

The profile introduces the agency layer into an ArchiMate repository by specialisation of existing element and relationship types, as the ArchiMate specification permits, so that existing tools display and exchange Charterion models without a language change. Stereotypes are named «charterion:…». Attributes are carried as properties. The projection of Section 6.4 is the identity on a repository that uses the profile. The table is generated from the toolkit's `archimate_profile.json`.

Table E.1. Charterion ArchiMate 3.2 specialisation profile: the Charterion elements and relations, their ArchiMate base element or relationship, stereotype, carried attributes, layer and reading.

Charterion element / relation	ArchiMate 3.2 base	Stereotype	Attributes carried	Layer / extension	Note
Agent	Application Component (active structure)	«charterion: Agent»	id; platform; charter refs	Application	Software actor that performs steps; never a Business Actor, because accountability stays with humans
Principal (human or role)	Business Actor / Business Role	«charterion: Principal»	vacancy flag	Business	Holds(h, r) is the standard Assignment from actor to role; a role without holders is vacant (WF1, WF9)
Charter	Contract (specialised Business Object)	«charterion: Charter»	id; level; state; rationale; validity from/until; review; cap permit/escalate/unit	Business	Associated to the Agent, to the scope element (capability, process or step) and to the Principal; the single origin of authority
Charter scope	Capability, Business Process or sub-process (step)	«charterion: Scope (association)»	scope kind	Strategy / Business	Scope resolution follows Realization (process realises capability) and composition (step in process)
Delegation	Association (directed) between two charterion:Agent components	«charterion: Delegation»	scope; declared level	Application	Effective level is the meet of the delegator's authority and the declared level (attenuation); WF6 checks the declaration
SoD guard	Constraint attached to two steps	«charterion: SoD»	step pair	Motivation	No agent may cover both steps (WF4)

continued on next page

(*continued*)

Charterion element / relation	ArchiMate 3.2 base	Stereotype	Attributes carried	Layer / extension	Note
Precedence guard	Constraint attached to two agents and a resource	«charterion: Precedence»	resource; a before b	Motivation	Resolves write contention on a shared resource (WF5)
Oversight	Association from Business Actor (or Role) to charterion:Agent	«charterion: Oversight»	none	Business / Application	Required for autonomous agents (WF7); counts towards the overseer's span
Escalation	Association from Business Actor (or Role) to charterion:Agent	«charterion: Escalation»	none	Business / Application	Human who decides an escalate outcome; defaults to Oversight, then to the charter principal (WF9)
Agentic step	Business Process (sub-process) or Business Function	«charterion: AgenticStep»	agentic = true	Business	Designated for agent performance; must be covered by an agent (WF2)
Reserved step	Business Process (sub-process)	«charterion: ReservedStep»	reserved = true; rationale	Business	Reserved to humans by business decision; no write-level authority may reach it (WF8)
Requires	Access relationship from step to resource	«charterion: Requires»	level	Business to Application / Technology	Level read maps to Access read; propose, commit and autonomous refine Access write
Resource	Application Component, Data Object, Application Service, Technology Service or Node	«charterion: Resource»	kind	Application / Technology	Anything a step needs and a grant can target
Grant	Association (directed) from charterion:Agent to resource	«charterion: Grant»	level; source system	Technology	Administered entitlement observed in IAM; reconciled with Need by WF3 (Shadow, Missing)
Need (derived)	Association (directed) from charterion:Agent to resource, derived	«charterion: Need»	level; derived = true; justifying steps	Application	Never authored; generated by the reasoner from Covers and Requires; feeds Bridge B1

continued on next page

(*continued*)

Charterion element / relation	ArchiMate 3.2 base	Stereotype	Attributes carried	Layer / extension	Note
Validity and Cap attributes	Properties of charterion:Charter (and charterion:Delegation)	profile attributes	valid from; valid until; review date; permit band; escalate band; unit	Business	Temporal profile drives WF10 (expired, stale); quantitative profile aligns with mandate bands (CC3)
Envelope assertion	Access or Association from charterion:Agent to resource, approved	«charterion: Envelope»	operation; level; approver; status; charter provenance	Application	Run-time authorisation rule of the execution plane; grounded by CC1, complete by CC2, approved by CC5
Mandate	Contract with an external Business Actor	«charterion: Mandate»	operations; counterparties; permit; escalate; unit; validity; approver	Business	Portable form of a charter for cross-organisational action; bounded by CC3, approver checked by CC5
Exposure	Application Interface or Business Interface offered to a counterparty	«charterion: Exposure»	counterparty; exposure level X0 to X4; status	Application / Business	External affordance inventory plus boundary envelope (Boundary plane)
Counterparty and trust weight	Business Actor (external) with property	«charterion: Counterparty»	trust weight in $[0, 1]$	Business	Effective confidence is confidence times trust weight (X-P3)
Assertion	Property set on any element or relationship of the slice	«charterion: Asserted»	source; confidence; validity; status (observed, approved, derived, attested)	All layers (cross-cutting)	Every fact of the substrate carries warrant; the Assertion Store is the model itself
Slice	Grouping	«charterion: Slice»	conformance level 0 to 4; span bound; parameters	All layers	Unit of application of the method and of the Conformance Statement (CC4)
Provisioning from the model (D5)	Work Package delivering Grants and Envelope assertions	«charterion: Provisioning»	charter refs; target plateau	Implementation & Migration	Bridge B1 as a planned change; the gap between Need and Grant is the work
Evidence Record	Deliverable attached to charterion:Charter	«charterion: Evidence»	reversed rate; denied rate; unwarranted share; escalation acceptance	Implementation & Migration	Bridge B2; gates evidence-based level promotion at charter lifecycle gates

Table E.3. ArchiMate 3.2 specialisation profile, additions in v1.1 (continuation of Table E.1); E14: carriers for IRREVERSIBLE and COMPENSATION

Charterion element / relation	ArchiMate 3.2 base	Stereotype	Attributes carried	Layer / extension	Note
Irreversible(x)	Application Component / Node (resource)	attribute on charterion:Resource	`irreversible`	Application / Technology	E14
Compensation(x, s_c)	Association resource → Business Process/Function	charterion:Compensates	—	Business	E14; **WF12**
Config(a, κ)	Application Component (agent)	attribute on charterion:Agent	`runningConfig`	Application	from AG-BOM
Reviewed(m, κ)	Contract (charter)	attribute on charterion:Charter	`reviewedConfig`	Business	G2
Instance(i, a)	Node / System Software	charterion:Instance	`type`, `since`	Technology	realises the Agent component
Cardinality(a, n)	Application Component	attribute on charterion:Agent	`maxInstances`	Application	
Performs(r, s)	Assignment Business Role → Business Process/Function	— (base relation)	—	Business	reserved steps
DelegationCap	Flow / Serving (delegation)	attributes on charterion:Delegation	`permitCap`, `unit`	Business	
Delegation validity	charterion:Delegation	attributes	`validFrom`, `validUntil`, `review`	Business	**WF10**
Obligation(o, σ, instr, kind)	Requirement / Constraint	charterion:Obligation	`instrument`, `clause`, `kind`	Motivation	Influence/Realization to the scope element; instrument as Driver
Collaboration(a, b)	Application Collaboration, or Flow	charterion:MayDelegate	—	Application	
BlastBound, AgentClass, ClassBlastBound	Application Component; Constraint	attributes; charterion:ClassBound	`blastBound`, `class`	Application / Motivation	
Authority certificate	Contract / Representation	charterion:Certificate	`claim`, `digest`	Business	realises the envelope entry

F Mapping to the TOGAF ADM

The table maps every phase and bridge of the Continuous Method to the TOGAF 10 ADM phase in which it is performed when an organisation runs both, the TOGAF deliverable it touches and the Charterion artefact it produces. Chapter 17 discusses the two habits that change. The table is generated from the toolkit's `togaf_mapping.json`.

Table F.1. Mapping of the Charterion Continuous Method (design loop D1–D5, operation loop O1–O6, boundary loop X1–X4, bridges B1–B2) to the TOGAF 10 ADM phases, the TOGAF deliverables touched and the Charterion artefacts produced.

Charterion phase	TOGAF 10 ADM phase	TOGAF deliverable / artefact touched	Charterion artefact produced
Set-up (precedes D1): agency governance	Preliminary	Architecture Principles; Tailored Architecture Framework; Architecture Governance Framework	Agency Governance Board charter; parameters (span bound, agency-debt weights); Conformance Statement template
D1 Populate the core	B Business Architecture; C Information Systems Architectures; D Technology Architecture	Architecture Definition Document; Business Capability Map; Process Flow diagrams; Application Portfolio and Data Entity catalogues; Technology Portfolio catalogue	MXF core section (humans, roles, capabilities, processes, steps, resources, requirements)
D2 Charter the agents	B Business Architecture (agency layer); C Information Systems Architectures	Actor/Role catalogue; Business Interaction matrix; Application/Function matrix	Agent Register; Charter Register; Delegation Register; Guard Register (SoD, Precedence, Oversight, Escalation); Principal and Vacancy Register
D3 Analyse	Requirements Management; Architecture Compliance review (B to D)	Requirements Impact Assessment; Compliance Assessment	Well-formedness Report (WF1 to WF12); Authority Matrix; Need-vs-Grant Matrix; Contention Matrix; SoD Matrix; Oversight Span Matrix
D4 Refine	B to D iteration; H Architecture Change Management	Architecture Definition Document (revised); Change Request	Charter Rationale Record; Accepted-Defect Register; revised Charter Register
D5 Provision from the model	E Opportunities and Solutions; F Migration Planning; G Implementation Governance	Implementation and Migration Plan; Transition Architecture; Architecture Contract	Grant Reconciliation Ledger; provisioned grants; approved envelope assertions (via B1)
O1 Observe	G Implementation Governance	Compliance Assessment; Implementation Governance Model	Assertion Store (observed assertions)
O2 Reconcile	H Architecture Change Management	Architecture Compliance review; Change Request	Assertion Store (approved assertions); Principal and Vacancy Register (current)
O3 Commit	G Implementation Governance	Architecture Contract	Approved envelope; ownership commitments
O4 Serve	G Implementation Governance (run time)	Architecture Contract (enforced through the tool/action gateway)	Decision Log (permit, deny, escalate, unwarranted outcomes)

continued on next page

(*continued*)

Charterion phase	TOGAF 10 ADM phase	TOGAF deliverable / artefact touched	Charterion artefact produced
O5 Account	G Implementation Governance; H Architecture Change Management	Compliance Assessment; Architecture Governance reporting	Evidence Record per charter; agent KPIs
O6 Detect drift	H Architecture Change Management	Change Request; Architecture Requirements Specification update	Agency Debt Dashboard; drift report; Coherence Report (CC1 to CC6)
X1 Declare exposure	A Architecture Vision; B Business Architecture	Stakeholder Map; Business Interaction matrix; Business Service/Interface catalogue	Exposure Declaration
X2 Issue mandates	G Implementation Governance	Architecture Contract with external parties	Mandate Register
X3 Transact under ADM cross	G Implementation Governance (run time)	Architecture Contract (both sides)	Decision Log (cross-boundary); Commitment Ledger
X4 Reconcile trust from commitments	H Architecture Change Management; Requirements Management	Compliance Assessment; Change Request	Trust Weight Register (updated)
B1 Bridge D5 to O3	F Migration Planning to G Implementation Governance	Implementation and Migration Plan handed to the Architecture Contract	Envelope assertions with provenance equal to the charter id; approver set equal to the accountable set
B2 Bridge O5/O6 to D4	H Architecture Change Management to Requirements Management	Change Request; Requirements Impact Assessment	Evidence Record feeding refine; evidence-gated promotion or degradation decisions

Table F.3. Mapping to the TOGAF ADM, additions in v1.1 (continuation of Table F.1)

Charterion phase / profile	TOGAF 10 ADM phase	TOGAF deliverable / artefact touched	Charterion artefact produced
staffing (D1)	B Business Architecture	Actor/Role catalog; Role/Process matrix	Principal & Vacancy Register, reserved-steps section
obligation (D1–D3)	Preliminary; A; G	Architecture Principles; Requirements; Compliance Assessment	Obligation facts; Regulatory Evidence Report
perimeter (D2)	C Application Architecture	Application Interaction matrix	Delegation Matrix, declared edges
cap-composition (D2)	C / E	Solution Building Blocks	Delegation Matrix, caps
configuration (G2, O6)	G Implementation Governance; H Change Management	Compliance Assessment; Change Request	Charter Register reviewed digests; drift events
instance (O1, G8)	G; H	Implementation governance; retirement	Agent Catalogue instances
containment (D3–D5)	D / E; G	Technology Architecture; security governance	Agent Catalogue bounds
certificate (D5, O2)	G; F Migration Planning	Compliance evidence; release plan (re-issue per release)	Envelope Register certificates; Audit viewpoint

G The Charterion Model Exchange Format

Charterion-MXF 1.0 is a JSON document validated by the JSON Schema (draft 2020-12) `charterion_mxf_schema.json` supplied with the toolkit. Every property carries a description; identifiers are strings unique within their sort; levels are the four literals; dates are ISO 8601. The validator checks the schema, then referential integrity (every step's process, every charter's agent, scope and principal, every requirement's step and resource exist; no step both agentic and reserved; approvers are humans), then runs the analysis at the requested instant. The table lists the sections.

Table G.1. The sections of the Charterion Model Exchange Format (Charterion-MXF 1.0), their content, the predicates and constructs they carry, and the checks that consume them.

MXF section	Content	Predicates / constructs carried	Consumed by
Header	Format version, model id, organisation, slice, reference date as_of	Slice identity; default evaluation instant t	All checks; report header
core	Humans, roles, holds, capabilities, processes, realizes, steps, before, agentic, resources, requires	Human, Role, Holds, Capability, Process, Realizes, Step, Before, Agentic, Resource, Requires	Design plane D1; WF2, WF4, WF5 (order), Covers and Need derivation
agency	Agents, charters, delegations	Agent, Charter(m, a, scope, level, principal), Delegation(a, b, scope, level)	Auth, Acc derivation; WF1, WF6, WF7; Bridge B1
agency.guards	Separation-of-duty pairs, precedence declarations, oversight	SoD, Precedence, Oversight	WF4, WF5, WF7, WF9 (span)
extensions	Reserved steps, escalation, temporal validity and review, quantitative caps, slice span bound	Reserved(s), Escalation(a, h), Validity(m, from, until), Review(m, review), Cap(m, permit, escalate, unit), SpanBound	WF8, WF9, WF10 (time-indexed model M_t), CC3 (cap), agency-debt AD_reserved and AD_stale
bridge	Technology-layer grants; envelope assertions with operation, level, approver and status	Grant(a, x, level); envelope entries (agent, operation, resource, level, approver, status, charter provenance)	WF3 (Shadow, Missing); CC1, CC2, CC5; Bridge B1
boundary	Mandates, exposure declarations, trust weights	Mandate (agent, operations, counterparties, permit, escalate, unit, validity, approver); Exposure (counterparty, level X0 to X4, status); trust weight t(B)	CC3, CC5; CC4 (exposure level); boundary loop X1 to X4
parameters	Organisation-set thresholds	theta_id, theta_act (ADM); SpanBound default	Execution-plane ADM configuration; WF9 default
metadata	Provenance of the file	created_by, created_at, tool, charterion_version, credit	Audit trail; Conformance Statement

Skeleton of a document

```
{ "charterion_mxf_version": "1.0", "model_id": "...", "organisation": "...",
  "slice": "...", "as_of": "2026-09-01",
  "core": { "humans": [], "roles": [], "holds": [], "capabilities": [],
```

```
          "processes": [], "realizes": [],
          "steps": [{"id": "...", "process": "..."}], "before": [],
          "agentic": [], "resources": [],
          "requires": [{"step": "...", "resource": "...", "level": "commit"}] },
  "agency": { "agents": [],
              "charters": [{"id": "...", "agent": "...", "scope": "...",
                            "level": "...", "principal": "..."}],
              "delegations": [{"from": "...", "to": "...", "scope": "...", "level": "..."}],
              "guards": { "sod": [], "precedence": [], "oversight": [] } },
  "extensions": { "reserved": [], "escalation": [],
                  "validity": [{"charter": "...", "from": "...", "until": "...",
                                "review": "..."}],
                  "caps": [], "span_bound": 12 },
  "bridge": { "grants": [],
              "envelope": [{"agent": "...", "operation": "...", "resource": "...",
                            "level": "...", "approver": "...", "status": "approved"}] },
  "boundary": { "mandates": [], "exposures": [], "trust_weights": [] },
  "parameters": { "theta_id": 0.8, "theta_act": 0.7, "span_bound": 12 },
  "metadata": { "created_by": "...", "created_at": "...", "tool": "...",
                "charterion_version": "1.0" } }
```

Table G.3. Charterion-MXF 1.1 sections added (continuation of Table G.1); all optional

MXF section	Content	Predicates / constructs carried	Consumed by
`extensions.irreversible`, `extensions.compensation`	irreversible resources; compensation steps	Irreversible, Compensation (E14)	**WF12**
`extensions.delegation_validity`	validity and review on delegations	temporal profile on Delegation	**WF10**
`extensions.configuration`	running and reviewed digests	Config, Reviewed	**WF13**
`extensions.instance`	instances, cardinalities	Instance, Cardinality	**WF14**
`extensions.staffing`	performer roles	Performs	**WF15**
`extensions.cap_composition`	delegation caps	DelegationCap	**WF16**
`extensions.obligation`	obligations	Obligation	**WF17**, evidence report
`extensions.perimeter`	collaboration edges, depth bound	Collaboration, δ	**WF18**
`extensions.containment`	blast bounds, classes	BlastBound, AgentClass, ClassBlastBound	**WF19**
`bridge.envelope[].certificate`	authority certificate per entry	Definition 11.1	**CC7**
`bridge.decision_log`	decision records with outcome and certificate	**Inv-5**, **CC6**, **CC7**	**CC6**, **CC7**
`conformance`	profiles in use; slices with execution levels; relationships with exposure levels	Conformance Statement items 2–3	**CC4**, vacuity guards

```
{ "charterion_mxf_version": "1.1",
  "metadata": { ... as in 1.0 ... },
  "core": { ... }, "agency": { ... }, "bridge": {
     "envelope": [ { "agent": "...", "resource": "...", "level": "commit", "status": "approved",
                     "certificate": { "cert_version": "1.1", "claim": {...}, "charter": {...},
                                      "chain": [...], "derivation": {...}, "accountable": {...},
                                      "model_digest": "sha256:...", "digest": "sha256:..." } } ],
     "decision_log": [ { "id": "r1", "agent": "...", "operation": "...", "resource": "...",
                         "outcome": "escalate", "t": "2026-...Z", "certificate": {...} } ] },
  "extensions": { "irreversible": [...], "compensation": [ {"resource": "...", "step": "..."} ],
     "delegation_validity": [...], "configuration": { "running": [...], "reviewed": [...] },
```

```
    "instance": { "instances": [...], "cardinality": [...] }, "staffing": { "performs": [...] },
    "cap_composition": { "delegation_caps": [...] }, "obligation": { "obligations": [...] },
    "perimeter": { "collaboration": [...], "depth_bound": 2 },
    "containment": { "blast_bound": [...], "agent_class": [...], "class_blast_bound": [...] } },
 "conformance": { "profiles": ["temporal", "reserved", "configuration", ...],
    "slices": [ { "id": "...", "scopes": [...], "execution_level": 3 } ],
    "relationships": [ { "id": "...", "slice": "...", "exposure_level": "X2" } ] } }
```

H Version policy

Charterion versions are numbered *major.minor*. A *minor* version may add optional predicates in a new stratum, new artefacts, new viewpoints, new mappings and new testable propositions, and may correct text; it SHALL NOT change the semantics of any existing predicate, condition or algorithm, so that a model conformant to $1.x$ is conformant to $1.y$ for $y > x$ at the same level. A *major* version may change semantics and requirements, and SHALL publish a migration note for models and Conformance Statements. Every construct is marked with the version that introduced it; the reference implementation's test models and expected outputs are the executable definition of each version. Errata are published against a version and incorporated at the next minor release. Proposals for change are addressed to the author; the intent stated in the Preface is that the ArchiMate profile and the method mapping be offered to The Open Group Architecture Forum for consideration as guides, and that the research constructs continue to be reported in the peer-reviewed literature.

New in Charterion v1.1

Version 1.1 under this policy. Version 1.1 is a minor version. It adds optional predicates in one new stratum (stratum 5), eight optional profiles with the conditions **WF13**–**WF19** and **CC7**, artefacts, viewpoints, mappings, MXF sections and testable propositions AP10–AP18, and corrects text (errata E1–E16, Appendix Q); it changes the semantics of no existing predicate, condition or algorithm, and requires no new condition at any conformance level. The mechanical form of the guarantee is the regression of Section 26.9: on every v1.0 model the v1.1 reasoner, with no v1.1 profile in use, reproduces the v1.0 results, and the v1.0 toolkit files are shipped unchanged. Candidates that would need a semantic change are recorded, with migration-impact notes, in Appendix J for a future major version.

I Comparative replication protocol for Table 24.2-bis

New in Charterion v1.1

This annex is the protocol under which Tables 24.2 and 24.3 were coded and under which a second analyst can replicate them (limitation 3). It is informative. The cell-level evidence—546 cells, each with mark, excerpt of at most 25 words, location, URL or DOI, retrieval status and adjudication note—is shipped as `40_annexes/table24_2bis_cells.csv`; the retrieval record and the adjudication record are in `40_annexes/AnnexY_landscape_verification.md`.

Unit of coding. One cell = (construct row, work column); one mark and one evidence excerpt per cell.

Marks. • *provides*: the work states the construct as a requirement (shall/must/should/required, or an enumerated control) *or* supplies a mechanism, data structure, schema, algorithm or protocol element that realises it, so that a reader could implement or check it from the text. ○ *names the concern*: the risk, goal, principle or question is mentioned without a construct. – *out of scope*: absent or explicitly excluded. The Charterion column is fixed by this specification and is not re-coded.

Evidence rule. Every • or ○ cell carries a verbatim excerpt of at most 25 words with its location (section, page, heading or JSON path), taken from the primary text; a – cell records at least three search terms and "no occurrence". If only a secondary source could be retrieved the cell is tagged [UNVERIFIED] and the mark cannot exceed ○. Nothing is coded from memory.

Retrieval status per work. *primary_full*: the full primary text was retrieved and read; *primary_excerpt*: the primary URL was confirmed and its text read only through search-engine renderings of that URL, because the site serves non-browser clients an error (no browser impersonation, mirror or cache is used); marks up to • but the cell says "(excerpt)" and the column carries [?]; *secondary*: [UNVERIFIED], mark $\leq$ ○.

Tie-break rules. (1) • versus ○: ○ unless the excerpt contains a normative verb or names a mechanism, field, schema or protocol message. (2) ○ versus –: ○ if the concern is addressed in a descriptive or normative sentence; – if it appears only in a reference list, a glossary or a quotation of another work. (3) A construct delegated to another named standard is ○ unless the work adds requirements on the delegated element. (4) A construct realised for a narrower object than Charterion's (a per-tool permission rather than a business-scope charter) is $\bullet^{\dagger}$, with the qualifier recorded, never silently.

Inter-rater procedure. Two coders code blind from the same source files and this protocol; the CSV is the exchange format. Agreement is reported per work as raw agreement and Cohen's κ over the three marks; disagreements are resolved by the tie-break rules and unresolved cells are reported as "split" with both excerpts. The second coder must not see the first coder's marks or the v1.0 table until after coding. *Status:* one AI-assisted first reading with a written adjudication record exists; the second, human reading has not been done.

Construct rows and work columns. Rows R1–R23 are the rows of Table 24.2 of v1.0; R24 (decision outcome beyond permit/deny), R25 (agent configuration inventory) and R26 (EA repository consulted by agents at decision time, with warrant or derived authority) are added. The twenty-one works, their versions, dates, canonical URLs or DOIs, retrieval status and licences are listed in the retrieval record; all were retrieved on 6 September 2026.

J Candidates for a major version (Annex Z)

New in Charterion v1.1

Each item below was assessed for v1.1 and set aside because it would change the semantics of an existing predicate, condition or algorithm, or because it failed the agent-specificity test. Each is recorded with the reason and a migration-impact note. Nothing here is specified.

Z.1 Level degradation on configuration drift.
Replacing the level of a drifted charter by $\mathrm{pred}(\ell)$ would compose with Definition 8.3 and change M_t. Impact: every model with the configuration profile would need re-analysis; envelopes derived from drifted charters would shrink.

Z.2 Instance-aware core.
Stating Charter, Auth and Acc on instances rather than types. Impact: every rule of Figure 7.2 is restated; MXF `agency` changes; all v1.0 models need an instance mapping.

Z.3 Quantitative semantics.
Amounts inside the least model (cumulative and per-period caps, arithmetic on Need). Impact: the semantics leaves Datalog with stratified aggregation for a richer logic; Proposition 7.4 must be re-proved; **CC3** becomes an inference.

Z.4 Resource partitions.
Refining **WF5** so that two writers on disjoint partitions of one resource are not in contention. Impact: Contention changes for every model with partition facts; v1.0 conformance at D2 may change. Interim in v1.1: acceptance code `disjoint-partitions`.

Z.5 Schedules and load-dependent authority.
Time-of-day and load windows on charters (limitation 7). Set aside on agent-specificity: schedules of authority are classical access control; they also require a calendar sort outside Section 6.1.

Z.6 Jurisdiction and location.
Data-residency constraints on resources and cross-jurisdiction delegation. Set aside on agent-specificity; expressible as a resource-attribute profile in a future version.

Z.7 Resource classification.
Sensitivity classes with write restrictions. Set aside on agent-specificity (mandatory access control); the agent-specific part—no autonomous write to a classified resource—is expressible today as an obligation of kind human_decision on the steps that require the resource (**WF17**).

Z.8 Partial order of levels.
Limitation 6; a different attenuation operator.

K Protocol bindings (informative)

New in Charterion v1.1

Charterion defines constructs, not wire formats. This annex names, for each construct that must cross into a runtime or identity system, the standard element retrieved in this version to which a binding can attach. Every binding is informative; the primary texts were retrieved on 6 September 2026 (Appendix I). Where a binding is stated as a possibility rather than an existing mapping, it says so.

Table K.1. Bindings from Charterion constructs to retrieved standard elements

Construct	Standard element	Binding
CONFIG(a, κ)	OWASP Agent Control Standard, AG-BOM: snapshot after `sessionStart`, `agbom/changed` diffs	κ = digest of the AG-BOM snapshot; O1 imports the snapshot, O6 consumes `agbom/changed`
INSTANCE(i, a)	IETF AIMS draft-klrc-aiagent-auth-03: agent identity (client identifier / SPIFFE ID)	type identity as the agent identifier; instance as a sub-identifier or session; posture attributes may carry κ
Decision log (**Inv-5**)	Agent Control Standard trace events (OCSF / OpenTelemetry extension); AIMS §11 audit fields	register fields of Appendix L named to match where a counterpart exists (`agent`, `operation`, `resource`, `outcome`, `t`)
Envelope entry, mandate	OAuth 2.0 token exchange; AIMS agent token; AIP holder-attenuable delegation blocks	the charter-derived level and caps as token scope and claims; a certificate digest may be carried as a claim (possibility, not an existing mapping)
Authority Request Protocol step 1	Agent Control Standard `ask` outcome and `intent_extension`; AIMS §10.7 CIBA	the runtime's ask channel submits the request; steps 2–5 remain in the design plane
Escalate outcome	Agent Control Standard `ask` routed to an approver	the approver is the human of ESC (**WF9**), supplied by the model
ADM outcome (Definition 9.3)	OpenID AuthZEN Authorization API 1.0 `decision` (boolean); XACML 3.0 four-valued decision	AuthZEN carries only `true`/`false`: *escalate* and *unwarranted* must travel in the response `context` and a binding SHOULD map both to `false` at the decision field (fail-closed); XACML's INDETERMINATE can carry *unwarranted* and an OBLIGATION can carry *escalate*
Snapshot digest d	(none retrieved)	a signature by the provisioning pipeline is required for authenticity; out of scope
Core facts (D1 import)	MCP servers of SAP LeanIX, Ardoq, Bizzdesign	read capabilities, processes, applications into `core`; write of charters only through G2

L Register schemas (informative)

The Envelope Register, the Mandate Register and the Decision Log that **CC1**–**CC7** read are given JSON Schemas in `31_schema/registers/` (`envelope_register.json`, `mandate_register.json`, `decision_log.json`), and the authority certificate in `31_schema/certificate.json`. An envelope entry carries `agent`, `resource`, `level`, `status`, optional `operation`, `approver`, validity, `certificate` and `review_flag`; a mandate carries the issuing organisation, the counterpart, scope, level, caps, validity, `delegation_cap` and the certificate of its grounding need; a decision record carries `id`, `agent`, `operation`, `resource`, `outcome` (permit, deny, escalate, unwarranted), `t`, optional `reversed`, the warrant used and the certificate. Field names follow the Agent Control Standard trace events and the AIMS audit fields where a counterpart exists (Appendix K). The schemas are the interface an organisation's runtime must export to claim D4; limitation 9.

M Regulatory Evidence Report (informative)

The report is generated by `regulatory_evidence.py` from a model with the obligation profile. For each Obligation$(o, \sigma, instr, kind)$ it computes: *human_decision*—the steps of σ, whether each is Reserved, its performer roles and holders (**WF15**), the agents with authority on it and their levels, and UnmetObligation; *record*—the decision-log records on the resources the steps of σ require, their outcome shares and their **CC6** attribution; *oversight*—the write-level agents active in σ, their overseers resolved to humans, **WF9** reachability and **WF14** span; *information*—the steps and agents in scope. Each obligation receives a verdict: satisfied by the model, not satisfied, or not assessed (information kind). The report also lists the reserved steps that no obligation grounds. The instrument and clause labels are the organisation's data; the framework does not read law. On the five cases the reports are `cases/regulatory_evidence_cases.md` (22 obligations per iteration).

N Adversarial assumptions (informative)

Charterion assumes: (A1) the adversary controls one or more agents entirely—by prompt injection, model compromise or credential theft—but not the repository, the reasoner, the identity system or the substrate's commit path; (A2) the identity system enforces the provisioned grants, so a compromised agent acts at most within Grant, hence (**WF3**) within Need; (A3) delegatees honour delegations only as recorded, so the transitive reach of a compromised agent is Blast(a); (A4) the humans of Oversight and Escalation are not compromised. Under A1–A4 the resources a compromise of a can change are bounded by Blast(a) and, with the quantitative profile, the amounts by its caps; **WF19** bounds Blast by design, **WF16** bounds the caps a compromised delegator can fan out, **WF18** bounds where it can delegate, and **WF13** detects a redeployment under a changed model. What the framework does not defend: compromise of the repository or of the provisioning pipeline, which are its trust anchors—**CC7** certificates let a verifier holding a trusted snapshot digest *detect* a repository that serves wrong authority, not prevent it; collusion of two compromised agents outside any recorded delegation (**WF11** covers recorded dependence only); and covert channels between agents. No adversarial evaluation has been performed (limitation 12).

O Reference trust estimator (informative)

Limitation 10 of v1.0 left $t_A(B)$ unspecified. This annex gives a reference estimator; it is not normative, and the boundary plane does not prescribe a trust model. Each reciprocal commitment closed by B towards A (X4) is an observation—kept or broken—at day τ_i. With a Beta prior (α_0, β_0) set by the exposure level and a half-life h,

$$\alpha = \alpha_0 + \sum_i w_i \,\mathrm{kept}_i, \qquad \beta = \beta_0 + \sum_i w_i \,(1 - \mathrm{kept}_i), \qquad w_i = 2^{-(t-\tau_i)/h}, \qquad t_A(B) = \frac{\alpha}{\alpha + \beta}.$$

Proposition O.1 (Properties of the estimator)**.** *(P1)* $t_A(B) \in (0, 1)$. *(P2) Adding a kept commitment never lowers* $t_A(B)$ *and adding a broken one never raises it—the monotonicity X-P3 requires of trust updates. (P3) With no observations* $t_A(B) = \alpha_0/(\alpha_0 + \beta_0)$*, the declared prior of the exposure level. (P4) The influence of an observation vanishes as its age grows.*

Proof. (P1) $\alpha, \beta > 0$. (P2) $\partial(\alpha/(\alpha+\beta))/\partial\alpha > 0$ and $\partial/\partial\beta < 0$. (P3), (P4) by substitution. The property (P2) is also checked executably by `trust_estimator.check_monotone`. □

A credible lower bound (a Beta quantile) may be used in place of the mean for κ_{eff}; the reference implementation (`trust_estimator.py`, standard library) computes the mean. AP18 states the proposition that this estimator improves on a static $t_A(B)$; no data exist.

P Open Group submission path (outline, informative)

The Preface states the intent to propose the ArchiMate specialisation profile and the ADM mapping to The Open Group. This annex outlines the path; no submission has been made. (1) Package: Appendix E with Table E.3 as an ArchiMate 3.2 specialisation profile in the form of the Open Group's profile examples—elements, stereotypes, attributes, allowed relationships, with the metamodel fragment of Figure 7.2's extensional layer; Appendix F with Table F.3 as an ADM extension in the form of the TOGAF Series Guides; MXF 1.1 as an exchange-format annex aligned with the Open Exchange Format's element and relationship identifiers. (2) Licensing: the profile and mapping would be offered under terms compatible with The Open Group's publication licences, distinct from the licence of this document, at the author's discretion (Appendix H). (3) Evidence required before submission: the field study of Section 26.12 in at least one organisation, and the human second reading of Appendix I. (4) Working-group route: the ArchiMate Forum for the profile, the Architecture Forum for the ADM extension; a white paper first, a Series Guide second. The outline is the author's; it is not an Open Group document and implies no endorsement.

Q Change log v1.0 → v1.1

New in Charterion v1.1

Every change from v1.0 to v1.1 is listed here with its kind under Appendix H: *addition* (optional predicates in stratum 5, artefacts, viewpoints, mappings, propositions), *correction* (text), or *completion* (toolkit implements what v1.0 specified). No semantic change. The Phase records (inventory, landscape verification, decision log, mandatory checks) are shipped in `40_annexes/`.

Additions. Stratum 5 with Proposition 8.4 and Corollary 8.2 (Section 8.10); profiles configuration (**WF13**), instance (**WF14**), staffing (**WF15**), cap-composition (**WF16**), obligation (**WF17**), perimeter (**WF18**), containment (**WF19**) (Sections 8.11–8.17, Table 8.3); authority certificates and **CC7** (Section 11.5, Table 11.2); resolutions (Table 13.2), gates (Table 18.4), computed regulatory evidence (Chapter 19, Appendix M), metrics (Chapter 20), artefacts and viewpoints (Table 21.3), MXF 1.1 (Table G.3), tooling (Chapter 22), Table 24.2-bis and the restated claim (Section 24.5), profile recommendations by level (Table 25.2), validation rows, injection study, scalability, regression, AP10–AP18, limitations after v1.1, field-study protocol and agenda (Chapter 26), the v1.1 profiles on the five cases with Tables 23.11–23.14, appendix additions (A–H) and the annexes I–P; bibliography entries for the eleven works added to the landscape.

Corrections (errata of v1.0). E1 Table 6.1: rows for Irreversible, Compensation. E2 Definition 7.4: "**WF8**–**WF12**". E3 stale ranges "**WF1**–**WF10**" / "**WF8**–**WF10**" replaced where they denote all extension conditions (Table 2.1, §5.4 box, Table 18.2, §21.5, Table F.1). E4 Table 5.1: **WF11**, **WF12** in the design row. E5 Table 23.7 superseded by Table 23.7-bis with the $\mathrm{AD}_{\mathrm{stale}}$ column. E6 §26.3 "nine". E7 Conformance Statement item 3: reversibility (and the v1.1 profiles). E8 Table 26.1 coherence row: toolkit tested **CC1**–**CC3**, **CC5** in v1.0. E9 duplicate bibliography entry removed. E10 Agent Control Standard entry corrected (version 0.1.1, public 27 May 2026, OWASP GenAI since 1 September 2026, licences); IMDA entry updated to version 1.5. E11 Table 23.9: witnesses rendered with their step. E12 indefinite article before "Charterion". E13 Preface: name and subtitle order. E14 carriers for Irreversible/Compensation (MXF 1.1, Table E.3). E15 Proposition 8.3 restated constructively with the recharter step and the cycle case. E16 §22.1: `charterion_separation.wf12` evaluates on Need; the D12 rule of Chapter 8 is on Auth and is what the v1.1 module computes. Also: SAIF marks corrected in Table 24.2-bis (no identity class or delegation construct on the SAIF pages); CSA Agent Identity Governance Framework date replaced by "v1, 2026 (page undated)"; NIST NCCoE authors added; AARM "STEP UP".

Completions. **WF10** on delegations, **CC4**, **CC6** implemented (Section 26.8); **WF11**, **WF12** computed on every run.

Unchanged. Figure 7.2, Algorithm 1, Definitions 8.3, 9.4, 9.5, 10.5, the lattice $\mathbb{L}$, the four-level vocabulary, the 24 requirements, the design levels D0–D4, the licence texts, the trademark notice and the attribution line.

Bibliography

Abiteboul S, Hull R, Vianu V (1995) Foundations of Databases. Addison-Wesley, Reading, MA

Amazon Web Services (2025) The Agentic AI Security Scoping Matrix: a framework for securing autonomous AI systems. AWS Security Blog, November 2025. https://aws.amazon.com/blogs/security/the-agentic-ai-security-scoping-matrix-a-framework-for-securing-autonomous-ai-systems/. Accessed 6 September 2026

Antunes G, Bakhshandeh M, Mayer R, Borbinha J, Caetano A (2014) Ontology-based enterprise architecture model analysis. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). ACM, pp 1420–1422. https://doi.org/10.1145/2554850.2555176

Azevedo CLB, Iacob ME, Almeida JPA, van Sinderen M, Pires LF, Guizzardi G (2015) Modeling resources and capabilities in enterprise architecture: a well-founded ontology-based proposal for ArchiMate. Information Systems 54:235–262. https://doi.org/10.1016/j.is.2015.04.008

Barbosa A, Santana A, Hacks S, von Stein N (2019) A taxonomy for enterprise architecture analysis research. In: Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019), vol 2. SCITEPRESS, pp 493–504. https://doi.org/10.5220/0007692304930504

Bernardi ML, Casciani A, Cimitile M, Marrella A (2024) Conversing with business process-aware large language models: the BPLLM framework. Journal of Intelligent Information Systems 62(6):1607–1629. https://doi.org/10.1007/s10844-024-00898-1

Bertino E, Ferrari E, Atluri V (1999) The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security 2(1):65–104. https://doi.org/10.1145/300830.300837

Boella G, van der Torre L, Verhagen H (2006) Introduction to normative multiagent systems. Computational & Mathematical Organization Theory 12(2-3):71–79. https://doi.org/10.1007/s10588-006-9537-7

Boissier O, Bordini RH, Hübner JF, Ricci A, Santi A (2013) Multi-agent oriented programming with JaCaMo. Science of Computer Programming 78(6):747–761. https://doi.org/10.1016/j.scico.2011.10.004

Calvanese D, Casciani A, De Giacomo G, Dumas M, Fournier F, Kampik T, La Malfa E, Limonad L, Marrella A, Metzger A, Montali M, Amyot D, Fettke P, Polyvyanyy A, Rinderle-Ma S, Sardiña S, Tax N, Weber B (2026) Agentic business process management: a research manifesto. Information Systems 140:102738. https://doi.org/10.1016/j.is.2026.102738

Castelfranchi C, Falcone R (1998) Towards a theory of delegation for agent-based systems. Robotics and Autonomous Systems 24(3-4):141–157. https://doi.org/10.1016/s0921-8890(98)00028-1

Ceri S, Gottlob G, Tanca L (1989) What you always wanted to know about Datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering 1(1):146–166. https://doi.org/10.1109/69.43410

Chan A, Salganik R, Markelius A, Pang C, Rajkumar N, Krasheninnikov D, Langosco L, He Z, Duan Y, Carroll M, et al (2023) Harms from increasingly agentic algorithmic systems. In: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23). ACM, pp 651–666. https://doi.org/10.1145/3593013.3594033

Chan A, Ezell C, Kaufmann M, Wei K, Hammond L, Bradley H, Bluemke E, Rajkumar N, Krueger D, Kolt N, et al (2024) Visibility into AI agents. In: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24). ACM, pp 958–973. https://doi.org/10.1145/3630106.3658948

Cloud Security Alliance (2026a) Agent Identity Governance Framework. CSA Lab Space, 27 March 2026 (labelled by CSA as unofficial, AI-assisted research). https://labs.cloudsecurityalliance.org/agentic/agentic-identity-governance-framework-v1/. Accessed 6 September 2026

Cloud Security Alliance (2026b) Agentic AI Autonomy Levels and Control Framework, version 2.0. CSA Lab Space, March 2026 (labelled by CSA as unofficial, AI-assisted research). https://labs.cloudsecurityalliance.org/wp-content/uploads/2026/03/

agentic-ai-autonomy-levels-control-framework-v2-csa-styled.pdf. Accessed 6 September 2026

Cloud Security Alliance (2026c) The Agentic Trust Framework: zero trust governance for AI agents. CSA blog, 2 February 2026. https://cloudsecurityalliance.org/blog/2026/02/02/the-agentic-trust-framework-zero-trust-governance-for-ai-agents. Accessed 6 September 2026

Crampton J, Khambhammettu H (2008) Delegation and satisfiability in workflow systems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT '08), pp 31–40. https://doi.org/10.1145/1377836.1377842

Crampton J (2005) A reference monitor for workflow systems with constrained task execution. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT '05). ACM, pp 38–47. https://doi.org/10.1145/1063979.1063986

Dantsin E, Eiter T, Gottlob G, Voronkov A (2001) Complexity and expressive power of logic programming. ACM Computing Surveys 33(3):374–425. https://doi.org/10.1145/502807.502810

Dennis JB, Van Horn EC (1966) Programming semantics for multiprogrammed computations. Communications of the ACM 9(3):143–155. https://doi.org/10.1145/365230.365252

El Hassani S (2026a) Enterprise models as execution substrate: agentic AI and the displacement of the human addressee of enterprise architecture. Research note, manuscript under review

El Hassani S (2026b) Do practitioners want the enterprise model to govern the agent? A survey of 150 practitioners on enterprise architecture as execution substrate for agentic AI. Manuscript under review

El Hassani S (2026c) The Substrate Framework: an enterprise architecture framework for organizations acted upon by agentic AI. Manuscript under review

El Hassani S (2026d) The boundary substrate: extending enterprise architecture to agent-to-agent business. Manuscript under review

El Hassani S (2026e) Agency as an architecture layer: a formal enterprise architecture framework for agentic-AI-driven enterprises. Manuscript submitted to the Journal of Intelligent Information Systems

El Hassani S (2026f) Embedding AI capabilities into enterprise architecture: an AI-enabled target architecture with an agentic extension. Zenodo. https://doi.org/10.5281/zenodo.22308981

Errico H (2026) Autonomous Action Runtime Management (AARM): a system specification for securing AI-driven actions at runtime. arXiv:2602.09433

Esteva M, Rodríguez-Aguilar JA, Sierra C, Garcia P, Arcos JL (2001) On the formal specification of electronic institutions. In: Agent Mediated Electronic Commerce: The European AgentLink Perspective. Lecture Notes in Computer Science, vol 1991. Springer, Berlin, pp 126–147. https://doi.org/10.1007/3-540-44682-6_8

European Parliament and Council (2024) Regulation (EU) 2024/1689 of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union L, 12 July 2024

European Union (2026) Regulation (EU) 2026/1744 of the European Parliament and of the Council amending Regulation (EU) 2024/1689 as regards simplification measures for artificial intelligence (Digital Omnibus on AI). Official Journal of the European Union, 24 July 2026

Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R (2001) Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security 4(3):224–274. https://doi.org/10.1145/501978.501980

Fill HG, Fettke P, Köpke J (2023) Conceptual modeling and large language models: impressions from first experiments with ChatGPT. Enterprise Modelling and Information Systems Architectures 18(3):1–15. https://doi.org/10.18417/emisa.18.3

Gabriel I, Manzini A, Keeling G, Hendricks LA, Rieser V, Iqbal H, Tomašev N, Ktena I, Kenton Z, Rodriguez M, et al (2024) The ethics of advanced AI assistants. arXiv:2404.16244

Garcia-Molina H, Salem K (1987) Sagas. In: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pp 249–259. https://doi.org/10.1145/38713.38742

Google (2025) Secure AI Framework (SAIF) 2.0: map of AI risks and controls for agents. https://saif.google/secure-ai-framework/saif-map. Accessed 6 September 2026

Gregor S, Hevner AR (2013) Positioning and presenting design science research for maximum impact. MIS Quarterly 37(2):337–355. https://doi.org/10.25300/misq/2013/37.2.01

Hübner JF, Sichman JS, Boissier O (2007) Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1(3/4):370–395. https://doi.org/10.1504/IJAOSE.2007.016266

Hammond L, Chan A, Clifton J, Hoelscher-Obermaier J, Khan A, McLean E, Smith C, Barfuss W, Foerster J, Gavenčiak T, et al (2025) Multi-agent risks from advanced AI. arXiv:2502.14143

Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Quarterly 28(1):75–105. https://doi.org/10.2307/25148625

Hinkelmann K, Gerber A, Karagiannis D, Thoenssen B, van der Merwe A, Woitsch R (2016) A new paradigm for the continuous alignment of business and IT: combining enterprise architecture modelling and enterprise ontology. Computers in Industry 79:77–86. https://doi.org/10.1016/j.compind.2015.07.009

Hu VC, Ferraiolo D, Kuhn R, Schnitzer A, Sandlin K, Miller R, Scarfone K (2014) Guide to attribute based access control (ABAC) definition and considerations. NIST Special Publication 800-162. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-162

Iacob ME, Jonkers H (2006) Quantitative analysis of enterprise architectures. In: Interoperability of Enterprise Software and Applications. Springer, London, pp 239–252. https://doi.org/10.1007/1-84628-152-0_22

Infocomm Media Development Authority (2026) Model AI Governance Framework for Agentic AI, version 1.5. Singapore; version 1.0 launched 22 January 2026, version 1.5 published 20 May 2026, updated 5 June 2026. https://www.imda.gov.sg/-/media/imda/files/about/emerging-tech-and-research/artificial-intelligence/mgf-for-agentic-ai.pdf. Retrieved 6 September 2026

ISO/IEC (2023) ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system. International Organization for Standardization, Geneva

Jonkers H, Lankhorst MM, ter Doest HWL, Arbab F, Bosma H, Wieringa RJ (2006) Enterprise architecture: Management tool and blueprint for the organisation. Information Systems Frontiers 8(2):63–66. https://doi.org/10.1007/s10796-006-7970-2

Lankhorst M (2017) Enterprise Architecture at Work: Modelling, Communication and Analysis, 4th edn. Springer, Berlin. https://doi.org/10.1007/978-3-662-53933-0

LePendu P, Dou D (2011) Using ontology databases for scalable query answering, inconsistency detection, and data integration. Journal of Intelligent Information Systems 37(2):217–244. https://doi.org/10.1007/s10844-010-0133-4

Li N, Mitchell JC, Winsborough WH (2002) Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. IEEE, pp 114–130. https://doi.org/10.1109/SECPRI.2002.1004366

Li M, Sun X, Wang H, Zhang Y (2012) Multi-level delegations with trust management in access control systems. Journal of Intelligent Information Systems 39(3):611–626. https://doi.org/10.1007/s10844-012-0205-8

Microsoft (2025) Process to build agents across your organization with Microsoft Foundry and Copilot Studio. Cloud Adoption Framework, Microsoft Learn, 3 December 2025. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ai-agents/build-secure-process. Accessed 6 September 2026

National Institute of Standards and Technology, National Cybersecurity Center of Excellence (2026) Accelerating the adoption of software and artificial intelligence agent identity and authorization. Concept paper, initial public draft, 5 February 2026. https://csrc.nist.gov/pubs/other/2026/02/05/accelerating-the-adoption-of-software-and-ai-agent/ipd. Accessed 6 September 2026

National Institute of Standards and Technology (2023) Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1

Norman TJ, Reed C (2010) A logic of delegation. Artificial Intelligence 174(1):51–71. https://doi.org/10.1016/j.artint.2009.10.001

OASIS (2007) Web Services Business Process Execution Language Version 2.0. OASIS Standard, 11 April 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Office of Management and Budget (2013) Federal Enterprise Architecture Framework Version 2. Executive Office of the President of the United States, Washington, DC

OWASP GenAI Security Project (2025) OWASP Top 10 for Agentic Applications for 2026. 9 December 2025. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/. Accessed 6 September 2026

OWASP GenAI Security Project (2026) Agent Control Standard (ACS), version 0.1.1 (specification pages 0.1.0). Public 27 May 2026; part of the OWASP GenAI Security Project since 1 September 2026. Apache-2.0 (specification, schemas, code), CC BY-SA 4.0 (documentation). https://github.com/GenAI-Security-Project/agent-control-standard. Retrieved 6 September 2026

Parasuraman R, Sheridan TB, Wickens CD (2000) A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 30(3):286–297. https://doi.org/10.1109/3468.844354

Pavlou PA (2002) Institution-based trust in interorganizational exchange relationships: the role of online B2B marketplaces on trust formation. The Journal of Strategic Information Systems 11(3–4):215–243. https://doi.org/10.1016/S0963-8687(02)00017-3

Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. Journal of Management Information Systems 24(3):45–77. https://doi.org/10.2753/mis0742-1222240302

Raza S, Sapkota R, Karkee M, Emmanouilidis C (2026) TRiSM for agentic AI: a review of trust, risk, and security management in LLM-based agentic multi-agent systems. AI Open 7:71–95

Rose S, Borchert O, Mitchell S, Connelly S (2020) Zero trust architecture. NIST Special Publication 800-207. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (1996) Role-based access control models. Computer 29(2):38–47. https://doi.org/10.1109/2.485845

Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T (2023) Toolformer: language models can teach themselves to use tools. In: Advances in Neural Information Processing Systems 36 (NeurIPS 2023), pp 68539–68551

Schneider FB (2000) Enforceable security policies. ACM Transactions on Information and System Security 3(1):30–50. https://doi.org/10.1145/353323.353382

Shneiderman B (2020) Human-centered artificial intelligence: reliable, safe & trustworthy. International Journal of Human–Computer Interaction 36(6):495–504. https://doi.org/10.1080/10447318.2020.1741118

Singh MP (1999) An ontology for commitments in multiagent systems: toward a unification of normative concepts. Artificial Intelligence and Law 7(1):97–113. https://doi.org/10.1023/A:1008319631231

Tamm T, Seddon PB, Shanks G, Reynolds P (2011) How does enterprise architecture add value to organisations? Communications of the Association for Information Systems 28:10. https://doi.org/10.17705/1CAIS.02810

Telang PR, Singh MP (2012) Specifying and verifying cross-organizational business models: an agent-oriented approach. IEEE Transactions on Services Computing 5(3):305–318. https://doi.org/10.1109/TSC.2011.4

The Open Group (2022a) ArchiMate® 3.2 Specification. Standard C226. The Open Group, Reading, UK

The Open Group (2022b) The TOGAF® Standard, 10th Edition. Standard C220. The Open Group, Reading, UK

U.S. Department of Defense (2010) DoD Architecture Framework Version 2.02. Office of the DoD Chief Information Officer, Washington, DC

Vasconcelos WW, Kollingbaum MJ, Norman TJ (2009) Normative conflict resolution in multi-agent systems. Autonomous Agents and Multi-Agent Systems 19(2):124–152. https://doi.org/10.1007/s10458-008-9070-9

Venable J, Pries-Heje J, Baskerville R (2016) FEDS: a framework for evaluation in design science research. European Journal of Information Systems 25(1):77–89. https://doi.org/10.1057/ejis.2014.36

Wang Q, Li N (2010) Satisfiability and resiliency in workflow authorization systems. ACM Transactions on Information and System Security 13(4):40. https://doi.org/10.1145/1880022.1880034

Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, Chen Z, Tang J, Chen X, Lin Y, Zhao WX, Wei Z, Wen J (2024) A survey on large language model based autonomous agents. Frontiers of Computer Science 18(6):186345. https://doi.org/10.1007/s11704-024-40231-1

Xi Z, Chen W, Guo X, He W, Ding Y, Hong B, Zhang M, Wang J, Jin S, Zhou E, et al (2025) The rise

and potential of large language model based agents: a survey. Science China Information Sciences 68(2):121101. https://doi.org/10.1007/s11432-024-4222-0

Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y (2023) ReAct: synergizing reasoning and acting in language models. In: The Eleventh International Conference on Learning Representations (ICLR 2023). arXiv:2210.03629

Zachman JA (1987) A framework for information systems architecture. IBM Systems Journal 26(3):276–292. https://doi.org/10.1147/sj.263.0276

Zaheer A, McEvily B, Perrone V (1998) Does trust matter? Exploring the effects of interorganizational and interpersonal trust on performance. Organization Science 9(2):141–159. https://doi.org/10.1287/orsc.9.2.141

Zambonelli F, Jennings NR, Wooldridge M (2003) Developing multiagent systems. ACM Transactions on Software Engineering and Methodology 12(3):317–370. https://doi.org/10.1145/958961.958963

OASIS (2013) eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, 22 January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. Retrieved 6 September 2026

OpenID Foundation (2026) Authorization API 1.0 (AuthZEN). Final, 11 January 2026. https://openid.net/specs/authorization-api-1_0.html. Retrieved 6 September 2026

Prakash (2026) Agent Identity Protocol (AIP). arXiv:2603.24775v1, 25 March 2026. https://doi.org/10.48550/arXiv.2603.24775. Retrieved 6 September 2026

South T, et al. (2025) Authenticated delegation and authorized AI agents. arXiv:2501.09674v1, 16 January 2025. https://doi.org/10.48550/arXiv.2501.09674. Retrieved 6 September 2026

draft-klrc-aiagent-auth-03: AI agent authentication and authorization (AIMS). Individual Internet-Draft, Informational, 6 July 2026. https://datatracker.ietf.org/doc/draft-klrc-aiagent-auth/. Retrieved 6 September 2026

Forrester Research (2026) The AEGIS framework: agentic AI enterprise guardrails for information security (public pages; report RES185394). https://www.forrester.com/technology/aegis-framework/. Page dated 16 April 2026; retrieved 6 September 2026

National Institute of Standards and Technology (2026) AI Agent Standards Initiative. Announced February 2026. https://www.nist.gov/artificial-intelligence/ai-agent-standards-initiative. Retrieved 6 September 2026 (page text through search-engine rendering)

SAP (2025) MCP server for SAP LeanIX solutions. Generally available 4 November 2025. https://help.sap.com/docs/leanix/ea/mcp-server. Retrieved 6 September 2026

Ardoq (2025) Using the Ardoq MCP server. Help Center article 415375; technology preview 9 May 2025, general availability September 2025. https://help.ardoq.com/en/articles/415375-using-the-ardoq-mcp-server. Retrieved 6 September 2026

Bizzdesign (2026) Model Context Protocol (MCP): turning enterprise architecture into AI-ready intelligence. Blog, 29 January 2026. https://bizzdesign.com/blog/model-context-protocol-mcp-turning-enterprise-architecture-ai-ready-intelligence. Retrieved 6 September 2026 (marketing pages, not documentation)

