

Contract-Bound Cognitive Routing

A Control Architecture for Agentic AI Integrity, Delegation, and Prompt-Injection Containment

CONSTRAIN. REASON. ASSURE.

Rob Brennan · Version 0.7 · 8 June 2026

DOI [10.5281/zenodo.20590723](https://doi.org/10.5281/zenodo.20590723) · bounded.uk/cbcr · CC BY 4.0

Abstract

Agentic AI systems connect probabilistic reasoning to tools, memory, external data, other agents, and state-changing operations. Their dominant failure mode – prompt injection – is an integrity problem: low-integrity input contaminating high-authority action. Contract-Bound Cognitive Routing (CBCR) treats it as one, modelling agentic execution as information flow over a typed, capability-gated graph mediated by a deterministic reference monitor, the MCP Policy Firewall.

This version organises the architecture around a single distinction: **what can be enforced deterministically, and what cannot**. A large class of agentic behaviour can be constrained by construction, near-airtight, with no reliance on the model's judgement – the control flow of a plan derived from trusted instructions, and structured data whose value-space is closed and validated. The boundary is precise, and it is the line most defences blur: schema validation checks shape, not meaning, so **type is not trust**. Beyond that line – free text, semantically-loaded fields, data-derived parameters, untrusted endpoints – lies a residual that cannot be made deterministic.

CBCR's contribution is to confine that residual to a single declared, fail-closed endorsement gate, make it measurable as false-labelling and false-endorsement rates weighted by reachable authority, and extend the discipline across multi-agent delegation through a monotonic non-amplification rule. It adopts the dual-path construction of Willison's dual-LLM pattern (2023) and CaMeL (Debenedetti et al., 2025); it does not solve conservative label propagation through a black-box model, which it states as the load-bearing open problem. As decentralized agent-discovery infrastructure – such as the Linux Foundation's DNS-AID – makes cross-organisation agent interaction routine, the population of untrusted endpoints and delegation chains grows, and an integrity-control layer of this kind becomes necessary rather than optional.

CBCR does not make untrusted content safe. It prevents untrusted content from reaching high-authority sinks except through a declared gate, and turns the risk left behind into a measured quantity. That is a smaller claim than "the firewall stops malicious payloads." It is also a stronger one.

1. Introduction

Agentic systems increasingly let a model read untrusted material – emails, web pages, tool responses, retrieved documents, messages from other agents – and then act: call tools, write memory, send messages,

change state. The recurring failure is prompt injection: content that was supposed to be *read* is instead *obeyed*, and the agent takes an action the attacker chose.

The architectural mistake underlying most attempts to fix this is to treat the model as though it can reliably enforce its own security boundaries. It cannot. A probabilistic system asked to recognise and resist adversarial instruction is exactly the wrong place to put a security guarantee. Enforcement has to live somewhere verifiable, outside the model.

CBCR is a proposal for that enforcement layer, and this paper is organised around what such a layer can and cannot promise. It proceeds in three movements, which are also the project's tagline:

- **Constrain** – the part you can enforce deterministically and verifiably: the control flow of the agent, and the structured data whose meaning is fully captured by its type. This is a large class, and within it the guarantee is near-airtight.
- **Reason** – the part you cannot: the probabilistic model itself, and the residual that survives because untrusted meaning cannot be soundly tracked through a black-box transformer.
- **Assure** – the response to that residual: confine it to one declared, fail-closed gate, bound its blast radius, and *measure* it rather than trust it.

Relationship to prior work and contribution boundary. CBCR adopts, rather than reinvents, the dual-path construction established by Willison's quarantined/privileged LLM pattern (2023) and formalised in CaMeL (Debenedetti et al., 2025): the component that processes untrusted content holds no capabilities, and the component that can reach sinks never sees raw untrusted text. That construction is taken as given. The contribution here is threefold: (i) the explicit deterministic/residual carve-out – naming where construction gives an airtight guarantee and exactly where it ends; (ii) a parametric residual-risk model that localises and measures what is left; and (iii) a monotonic non-amplification rule that extends the discipline to multi-agent delegation, checked against proof-carrying contract chains. The open problem – conservative label propagation through a black-box model – is inherited, not solved; it is stated and localised. The security property in Part III is an informal architectural adaptation of noninterference, not a theorem proven over a formal semantics, and it is presented as such.

2. Problem Statement

The root cause of prompt injection is structural: trusted instructions and untrusted data are concatenated into a single token stream, and there is no reliable way to make a model act on the first while merely reading the second.

The concern is **integrity, not confidentiality**. The danger is not that secrets leak; it is that low-integrity data contaminates high-integrity decisions. Named correctly – as a Biba-style integrity problem – decades of systems-security discipline become applicable: reference monitors, capabilities, and information-flow control.

By 2026 this was no longer theoretical. The Cloud Security Alliance characterised MCP security as a systemic crisis – among the documented incident classes are tool-description poisoning, cross-server tool shadowing, and the exfiltration of data by privileged agents acting on attacker-supplied content – and

recommended that agent tool interactions be mediated through an explicit policy layer rather than the agent's own reasoning [10]. CBCR is a formal architecture for exactly that layer.

Two definitions carry the rest of the paper:

- A **source** is where data enters the system (untrusted input: user prompts, retrieved content, tool responses, agent messages, web content).
- A **sink** is an exit point where information becomes a consequential effect: tool invocation, memory write, external redistribution, or re-entry into trusted reasoning context. Each sink carries a *required minimum integrity*.

Prompt injection is harmless until injected content reaches a sink. The whole architecture is therefore about controlling what is allowed to reach one.

Part I – Constrain

What can be enforced deterministically, and where that ends.

3. Enforcement must be verifiable, not the model

CBCR's first principle is to confine policy enforcement to a deterministic, verifiable **reference monitor** – the MCP Policy Firewall – and to keep that enforcement out of the model entirely. The monitor must hold the three classical properties (Anderson, 1972): it is **always invoked** (complete mediation), **tamper-resistant**, and **small enough to be verified**. This is the load-bearing assumption, not a metaphor.

"Deterministic" here is standing in for *verifiable*: a non-deterministic but formally verified enforcer would serve equally. Determinism is simply the cheapest route to verifiability. The corollary discipline is that the reasoning layer never enforces and never holds authority by virtue of what it says; **authority is held, not inferred from identity or from the model's confidence**. Authority is object-capability style: a contract grants a specific set of capabilities (routes, tools, sinks), and holding a capability is necessary but never sufficient, because an integrity requirement applies independently.

A firewall decision evaluates at least:

```
source_label
object_label
agent_contract
requested_route
requested_sink
sink_required_integrity
capability_grant
endorsement_status
delegation_chain
audit_requirement
```

A permitted transition requires **both** capability authorisation **and** integrity sufficiency. Either failure blocks the route. Everything the firewall checks is a label or a capability – never the meaning of content. That content-blindness is deliberate, and it defines the limit of Part I.

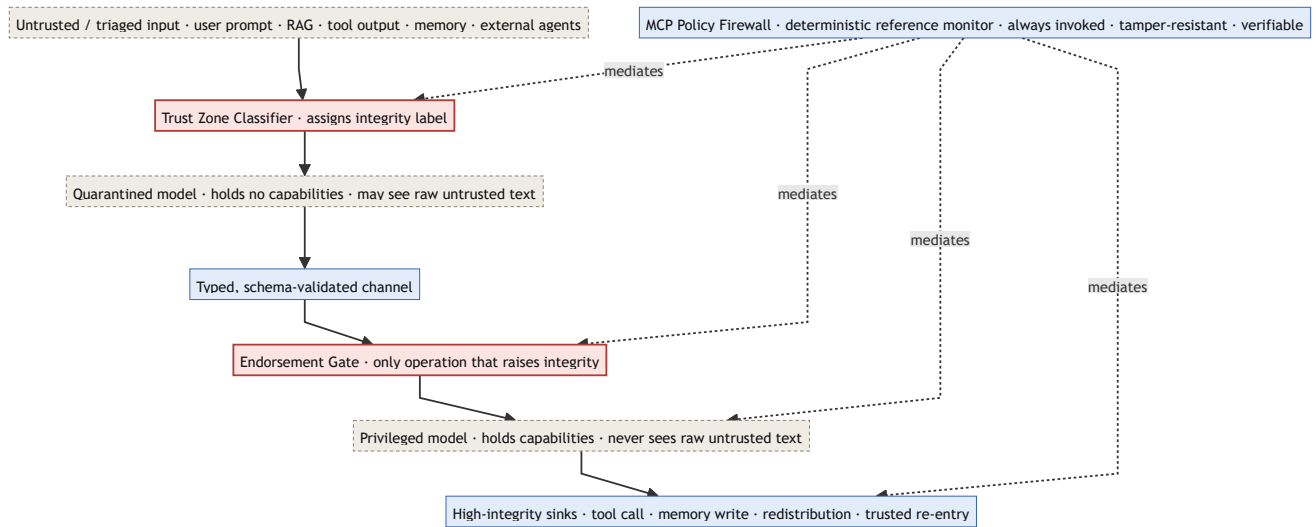


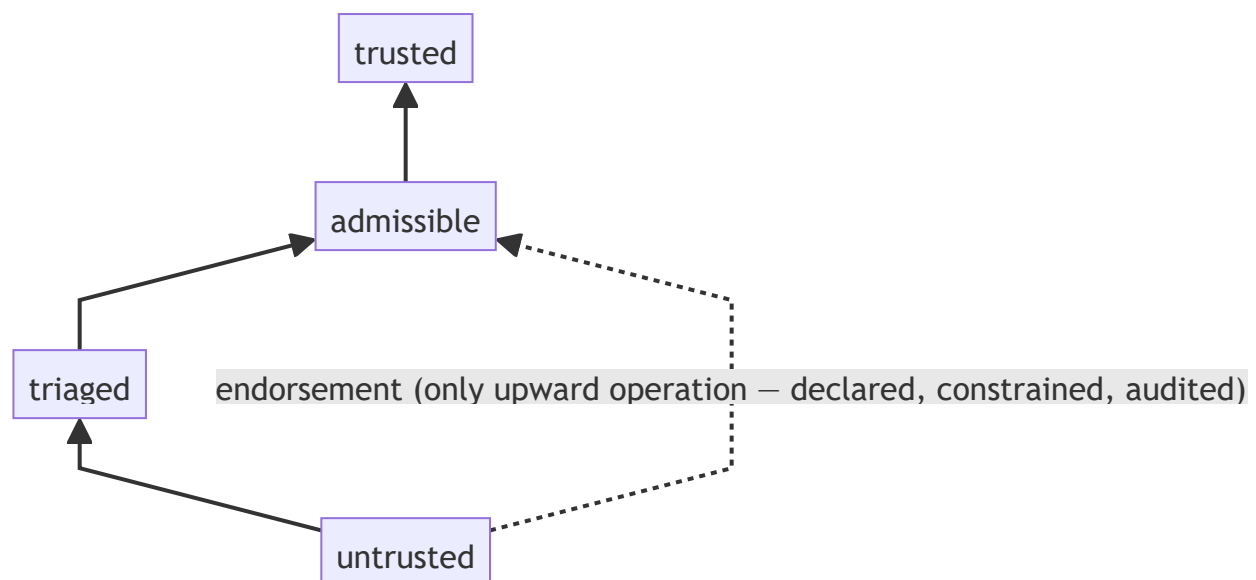
Figure – System overview. Blue is deterministic and verifiable (Part I); red marks the two probabilistic gates, the residual-risk loci (Part III); grey is probabilistic but contained by construction (Part II). The architecture does not rely on the grey components being correct.

4. The integrity model

Each object carries an integrity label drawn from a lattice, ordered low to high:

`untrusted` $\sqsubseteq$ `triaged` $\sqsubseteq$ `admissible` $\sqsubseteq$ `trusted`

`triaged` means an object has been typed, routed, or classified by policy but not yet endorsed as safe to influence a high-integrity sink. It is an integrity status, not a confidentiality classification. The governing rule: **lower-integrity content must not influence a higher-integrity sink except through an explicitly declared endorsement**, and the only operation that may raise an object's integrity is endorsement.



Sources are labelled on entry by the Trust Zone Classifier. Sinks carry a required minimum integrity. The firewall enforces, for every transition, that the object's integrity meets the sink's requirement and that the agent holds the capability.

5. What can be enforced deterministically — the controllable core

This is the part most treatments leave implicit, and it is where the strongest guarantee lives. Two large classes of agentic behaviour can be constrained *by construction*, with no reliance on the model's judgement.

5.1 Control-flow conformance

If the agent's plan — which tools it calls, in what order, against which sinks — is derived from the *trusted* instruction and pinned as a contract, the firewall can enforce conformance to it and block any deviation. Injected content that tries to make the agent *do something different* — call an unlisted tool, reach an unpermitted sink, change the plan — never executes, because the route does not exist in the contract. This is control-flow integrity, and it is the verifiable backbone of the architecture. An attacker can repeat instructions in untrusted text all they like; they cannot move the agent off a plan the firewall holds.

The condition is that the plan is generated from a trusted source. The moment a plan adapts to untrusted data — data-dependent control flow — the baseline itself becomes attacker-influenceable, and the dual-path discipline of Part II is required to keep planning sound.

5.2 Closed-value-space structured data

Where the agent collects *defined and specific* data — a request whose parameters come from a trusted source, returning a response whose value-space is closed and validated (an integer in range, one of N enum values, an allowlisted identifier, a format with a strict validator) — the firewall can validate it deterministically. Content that violates the structure is rejected outright. In this regime the "endorsement"

is not a fallible judgement at all; it is a schema or allowlist check, and "anything else crafted does not match" is literally true.

5.3 The combined guarantee

Within the controllable core – conforming control flow plus closed, validated structured data, from trusted parameters and trusted endpoints – CBCR's guarantee is near-airtight and requires no trust in the model. The engineering goal is to push as much of any system as possible into this regime. The rest of the paper is about the part that cannot be pushed there.

6. Where determinism ends: type is not trust

The deterministic guarantee fails at a precise and narrow line, defined by three conditions all having to hold:

1. **Parameters from a trusted source.** If a request parameter is derived from untrusted data ("look up the record named in this email"), the attacker cannot change the call's structure but can change its target.
2. **A trusted endpoint.** A well-shaped response from a compromisable third-party server is still untrusted input – schema-conformant untrusted input.
3. **A closed, meaning-bearing value-space.** This is the decisive one. **Schema validation checks shape and type, not meaning.** A field typed `string` validates no matter what it contains, so `{recipient_email: "string"}` passes identically whether it holds the intended address or the attacker's. The crafted value did not break the structure – it *is* the structure, with hostile content. **Type is not trust.**

When any condition fails, the data crosses out of the controllable core and into the residual: free text, semantically-loaded string fields, data-derived parameters, untrusted endpoints. That residual is the subject of Parts II and III. Stating the boundary explicitly is what lets a reader see that the bounded residual is a precise carve-out, not the whole story.

Part II – Reason

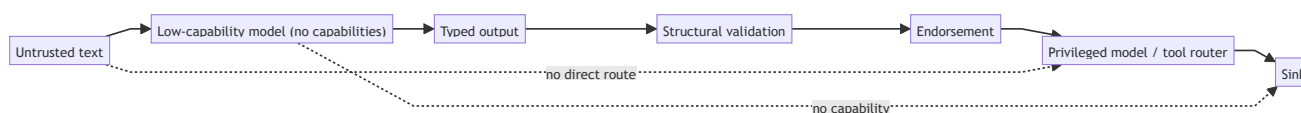
The contained probabilistic model, and the open problem at its core.

7. Construction over detection: the contained model

The residual is handled first by construction, not detection – the dual-path discipline adopted from the dual-LLM pattern and CaMeL:

1. The model instance that processes untrusted content holds no high-integrity capabilities; no high-integrity sink is reachable from it. It may emit only into typed, schema-constrained channels.
2. The privileged instance that can reach sinks never sees raw untrusted text. It consumes only typed

data that has passed structural validation and, where required, endorsement.



Undetected instruction content cannot reach a sink because the path from an untrusted context to a sink does not exist – not because an inspector caught it. Detection becomes defence-in-depth over a smaller surface, not the primary claim.

But construction does not escape the open problem of §8; it relocates and shrinks it. The dual-path discipline removes control-flow injection. What it cannot remove is the residual in the *data* the privileged instance consumes: the moment a typed channel carries a free-text field – a summary, an extracted value, a judgement – that the validator cannot semantically check, that field is the propagation problem wearing a schema. That irreducible surface is exactly the §6 residual, and it is where the endorsement gate of Part III lives.

8. The open problem: conservative label propagation

Standard information-flow control propagates labels through known transformations. An LLM is a black-box transformer: when it reads untrusted content and emits a summary, no static analysis can track the flow through its weights. The summary is derived from untrusted input but carries no automatic label.

The only **sound** rule available at the architecture level is conservative:

Any model output produced in a context that contained untrusted content is itself untrusted until endorsed.

This is sound but costly – it pushes large volumes of derived content through endorsement, the very surface the architecture wants to minimise. The central design tension is:

soundness of propagation $\rightleftharpoons$ endorsement load

CBCR does not solve this; it states it. Partial mitigations reduce endorsement load without weakening soundness – typed channels, provenance metadata, structural separation, narrow schemas, evidence objects rather than free-form text, capability-free untrusted processing, privileged reasoning that consumes only validated artefacts – but none eliminates it. Anyone who claims to have closed this should be asked for their propagation proof.

Part III – Assure

Confining the residual, bounding it, and measuring it rather than trusting it.

9. The endorsement gate

Endorsement is the only operation that may raise an object's integrity, so it is the entire residual-risk surface. It is also the **maximally adversarial** point in the system: the gate reads attacker-controlled content in order to decide whether to trust it, so the poisoned data is, in effect, arguing for its own promotion. If the gate is an LLM-judge it is vulnerable to the same injection it is adjudicating. There is no sound verification here; the gate guesses, well or badly, with no ground truth to check against.

Three disciplines make that tolerable without pretending it is solved:

- **Fail closed.** Uncertainty resolves to refuse, never to endorse. Endorsement may only reduce authority or pass; it must never grant authority. Endorsement is never implicit – it does not happen because the model paraphrased, summarised, or appeared confident.
- **Minimise the semantic surface.** Push as much of each endorsement decision as possible back into deterministic checks (schema, provenance, allowlist, range, format). Every part expressible as a deterministic predicate leaves the fallible gate and re-enters the controllable core of Part I. The ideal endorsement is almost entirely structural with a vanishing residue of genuine judgement.
- **Measure, do not trust.** The gate is not assumed sound; its error rate is estimated by adversarial red-teaming and by auditing every real endorsement.

Every endorsement record should capture:

Field	Purpose
Endorsement ID	Stable record identifier
Object ID	Object being endorsed
Previous label	Original integrity label
New label	Raised integrity label
Gate name	Endorsement mechanism used
Evidence	Evidence supporting endorsement
Authority reachable	Highest authority reachable after endorsement
Reviewer or service	Who or what endorsed it
Timestamp	When endorsement occurred
Confidence	Evidence quality or test basis
Audit result	Pass, fail, exception, override

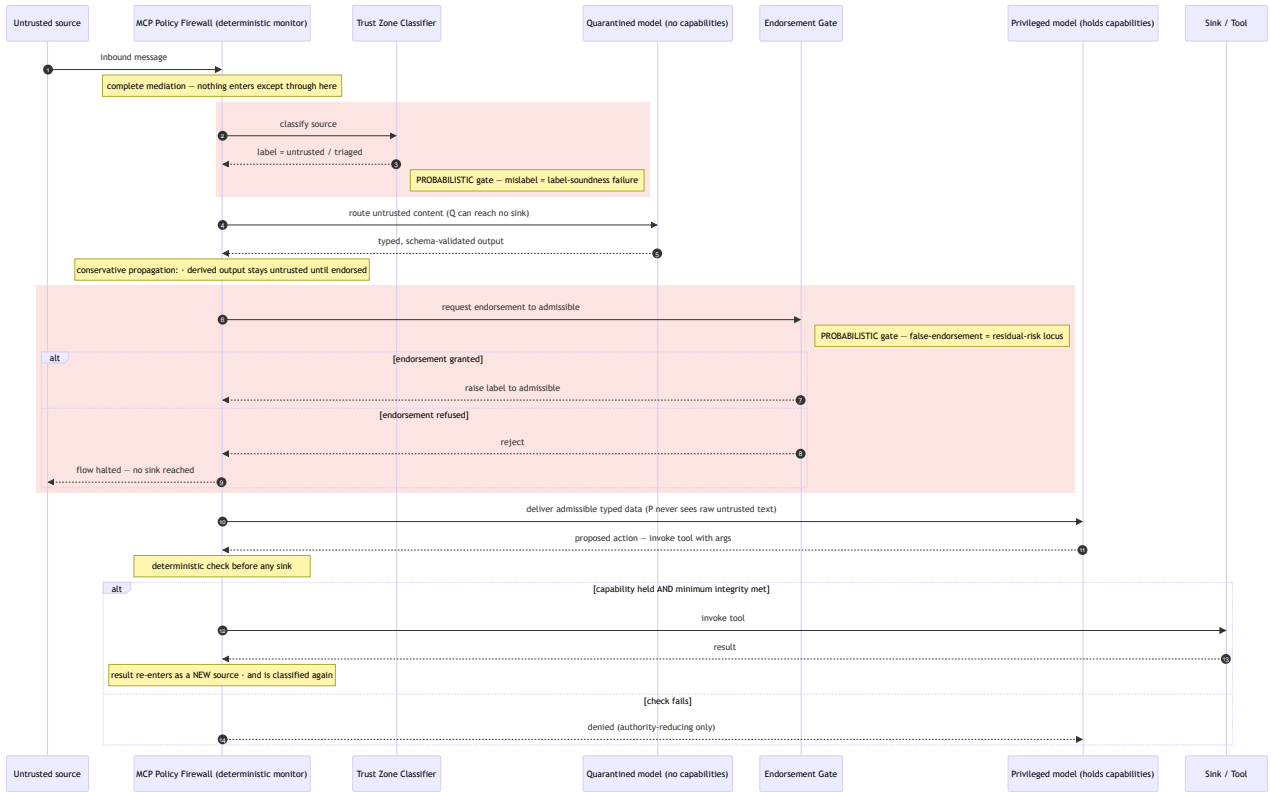


Figure – Request lifecycle. There are exactly two halt points – a refused endorsement and a failed capability/integrity check – and neither depends on catching malicious content. The privileged instance appears only after the gate; the quarantined instance never shares an exchange with a sink. The separation is a property of message order, not of inspector vigilance. The re-entry loop is where the $\S 8$ propagation cost compounds on chatty tool use.

10. The security property and the residual-risk model

Integrity-Constrained Mediation. Under correct contracts and a sound enforcement layer, no **untrusted** -labelled source may influence a high-integrity sink except through a declared endorsement gate. Equivalently: every permitted flow from untrusted input to an action is mediated by an endorsement operation the policy named in advance.

This is noninterference *modulo endorsement*. It is stated as a design property the architecture is built to satisfy, not a theorem proven over a formal semantics: "influence" is used in its operational sense, and no transition system or attacker model is given here. The value of stating it precisely is the falsifiable risk model that follows.

The system's residual integrity risk is the unsoundness of its **two probabilistic gates** – the Trust Zone Classifier that assigns labels on the way in, and the endorsement gate that raises them on the way out:

Residual integrity risk $\approx \Sigma_{\text{labels}} (\text{false-labelling rate} \times \text{authority of reachable sink})$
 $+ \Sigma_{\text{gates}} (\text{false-endorsement rate} \times \text{authority of reachable sink})$

This is a risk model, not a proof of safety. Its value is that it converts residual risk from prose scattered across the design into two measurable quantities attached to two named components, and it makes the claim falsifiable: measure the false-labelling and false-endorsement rates under adversarial input, measure the authority reachable past each gate, and the residual integrity risk the architecture leaves behind becomes testable. The deterministic core of Part I contributes nothing to this sum – by construction, it has no probabilistic failure term.

11. Capability contracts and multi-agent attenuation

A contract makes authority explicit and enforceable. A minimal contract includes:

Field	Purpose
Contract ID	Stable identifier
Agent identity	The agent or service bound to the contract
Permitted sources	Sources the agent may read
Permitted routes	Allowed graph transitions
Permitted tools	Tool capabilities granted
Permitted sinks	Sinks reachable under the contract
Required integrity	Minimum integrity label for each sink
Endorsement gates	Named gates that may raise integrity
Delegation rights	Whether the agent may delegate, and limits
Logging requirements	Events that must be recorded
Expiry and revocation	Contract lifetime and revocation mechanism

For multi-agent systems, CBCR applies a **monotonic non-amplification** rule:

If agent A delegates to agent B, then B's effective capabilities $\subseteq$ A's, and the sinks reachable by B $\subseteq$ those reachable by A.

Trust cannot grow along a chain. Each hop carries its contract chain as **proof-carrying delegation**, and the firewall verifies attenuation at every step rather than trusting a downstream agent's self-description. This bounds the blast radius of a compromised mid-chain agent to a subset of what the chain origin already held.

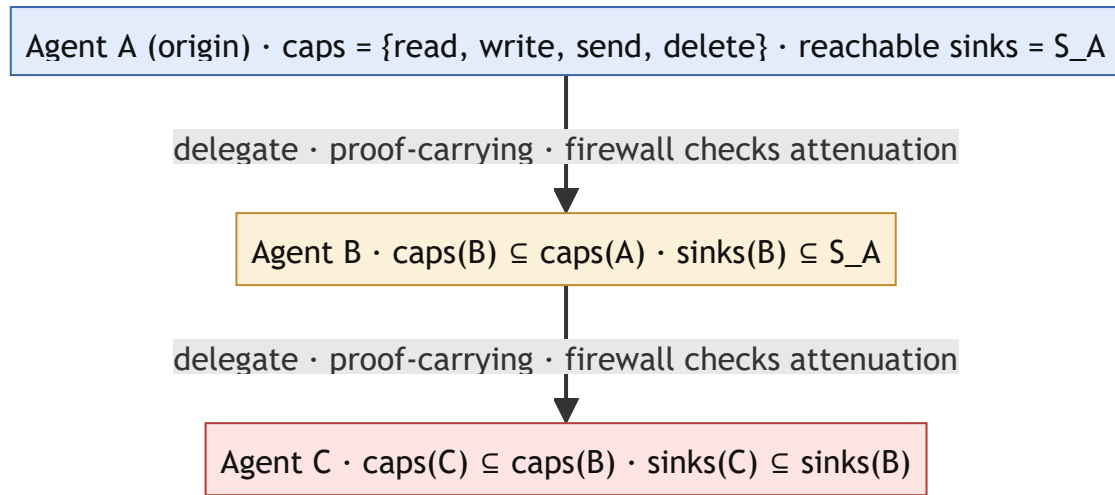


Figure – Capability attenuation. Authority can only shrink along a delegation chain; the firewall verifies the subset relation at each hop from the proof-carrying contract chain, bounding the blast radius of a compromised mid-chain agent.

The limit is worth naming: attenuation bounds **authority, not information**. It governs what a compromised agent can *do*; it does not govern what it can *say* to its delegees, where the propagation problem of §8 still applies down the chain. The two concerns are orthogonal, and this rule addresses only the first.

Composition with decentralized discovery. This matters increasingly because agents are beginning to find and address one another at runtime rather than through hardcoded integrations. The Linux Foundation's DNS-AID project (2026), for instance, lets agents and MCP servers be published, discovered, and verified over existing DNS infrastructure. Such infrastructure sits *below* CBCR and composes with it: discovery and identity answer *who* an agent is and *where* to reach it, and even cryptographic agent verification establishes authenticity, not content-safety. CBCR governs the layer above – *what* a discovered agent is authorised to do, and how its content is kept from reaching a sink – consistent with the principle (§3) that authority is held, not inferred from identity. Decentralized discovery therefore enlarges precisely the surface this section governs: every newly-discovered counterpart is, by the criteria of §6, an untrusted endpoint until contract and integrity say otherwise. A discovery substrate that already publishes capability descriptors is also a natural place to publish and resolve CBCR contracts, though CBCR depends on no particular one.

12. Evaluation strategy

CBCR should be evaluated against its own claims, not a broader promise it does not make.

Complete mediation. Can any source reach any sink without crossing the firewall? Enumerate all tool, memory-write, external-communication, and agent-to-agent routes; test direct-API, plugin, and connector bypasses; verify denials are logged. Metric: percentage of source-to-sink paths mediated. Target: 100%; anything lower voids the core property.

Deterministic-core conformance. Does pinned control flow reject deviation, and does closed-value-space

data reject structurally-invalid responses? Metric: deviation- and malformed-input rejection rate. Target: 100% within the closed regime.

Endorsement false-positives. How often does the gate raise malicious or low-integrity content to admissible? Test with adversarial prompt-injection, RAG-poisoning, malicious-tool-response, encoded-payload, role-confusion, and multi-hop-delegation corpora. Metric: false-endorsement rate by gate, input class, and sink authority.

Reachable authority. After endorsement, what can the content influence – which tools, data classes, memory writes, external channels, state changes, downstream agents? Combined with the rate above, this yields the residual-risk figure.

Label propagation. Does derived model output retain conservative integrity status across summarisation, structuring, mixing with trusted context, planning, memory-candidate creation, and reintroduction through a downstream agent? Expected rule: any output produced in a context containing untrusted input remains untrusted until endorsed.

13. Governance and assurance

CBCR turns vague behavioural assurance into a set of concrete artefacts to review: contract inventory, capability register, route map, source-to-sink graph, integrity-label policy, endorsement-gate design, false-endorsement test results, reachable-authority analysis, delegation-chain records, firewall-bypass tests, control-plane protection evidence, and audit-log samples.

The review question becomes a checklist: Are all source-to-sink paths mediated? Are labels soundly assigned? Are integrity transitions explicit? Are capabilities bounded? Are delegation chains attenuating? Are endorsement gates measured? Is the residual risk accepted by an accountable owner?

This is the "Assure" movement: the security posture is not a behavioural promise but a set of measured quantities and auditable records an accountable owner can sign off.

Relationship to industry risk frameworks. These artefacts also map onto the assurance frameworks security teams already use, which eases adoption. CBCR's source-to-sink graph and capability register populate the tool-manipulation and supply-chain threat categories of the Cloud Security Alliance's MAESTRO threat model for agentic systems [10]; its reference monitor, label policy, and attenuation rule instantiate control objectives in the AI Controls Matrix for input validation, least-privilege access, and runtime mediation; and its founding principle – authority is held, not inferred (§3) – is the zero-trust maxim *never trust, always verify* applied at the tool-integration layer. CBCR is best understood not as an alternative to these frameworks but as the concrete architecture that satisfies the control objectives they specify.

Back Matter

14. Validity assumptions

The security property holds if and only if:

1. **Complete mediation.** No source-to-sink path bypasses the firewall.
2. **Label soundness.** The Trust Zone Classifier labels sources correctly; its error rate is the first term of the residual-risk model.
3. **Control-plane integrity.** Contracts, registries, and the firewall are uncompromised; the property says nothing under a compromised monitor.
4. **Conservative propagation.** Integrity labels propagate through the reasoning layer without being laundered (§8).

Stating these is itself a validity gain: the guarantee is conditional on explicit, testable assumptions rather than asserted unconditionally.

15. Threat model

In scope

Threat	CBCR response
Prompt injection	Separation, labels, endorsement, no direct sink route
Tool misuse	Capability check plus sink integrity requirement
Memory poisoning	Memory write treated as a high-integrity sink
RAG poisoning	Retrieved content enters low-integrity unless endorsed
Delegation abuse	Capability attenuation and proof-carrying delegation
Output laundering	Conservative propagation rule
Control-plane compromise	Out of property scope; must be separately protected

Out of scope / not solved

Semantic safety of endorsed content; malicious insiders with control-plane access; model hallucination unrelated to untrusted-to-sink flow; confidentiality leakage except where treated as a sink policy; all downstream misuse of correctly endorsed information; full static information-flow tracking through model weights.

16. Relationship to existing defences

CBCR repositions existing mitigations rather than replacing them.

Defence	CBCR position
Input filtering	Defence-in-depth, not primary control
Prompt hardening	Useful but not enforcement
Instruction hierarchy	Useful but model-internal and fallible
Semantic inspection	Endorsement support, not a complete boundary
Human approval	Strong endorsement gate if scoped and auditable
Schema validation	Deterministic control for the closed-value-space core (Part I)
Capability restriction	Core architectural control
Dual-model separation	Adopted construction (Willison, 2023; CaMeL, Debenedetti et al., 2025); CBCR's role is to carve out the deterministic core, measure the residual, and extend to multi-agent chains
Decentralized agent discovery (e.g. DNS-AID)	Composes <i>below</i> CBCR: discovery, addressing, and identity verification – not authorization or integrity. It enlarges the untrusted-endpoint surface CBCR governs

The central shift: safety is not claimed because the model recognises malicious content. It is claimed deterministically where architecture closes the value-space, and only as a *measured residual* where it cannot.

17. Limitations

CBCR does not make the meaning of endorsed content safe; it controls the path by which content can influence action. It depends on complete mediation, label soundness, and control-plane integrity – any bypass or compromise voids or weakens the property. Conservative label propagation through LLMs remains the load-bearing open problem. Endorsement gates are fallible; CBCR makes that fallibility measurable, it does not remove it.

18. Claims

1. **Deterministic core.** Conforming control flow and closed-value-space structured data (from trusted parameters and endpoints) are enforced by construction, with no reliance on model judgement.
2. **Mediation.** CBCR enforces integrity-constrained mediation: every permitted untrusted-to-action flow must cross a declared endorsement gate.
3. **A precise boundary.** The residual is exactly the data that resists closure – free text, semantically-

loaded fields, data-derived parameters, untrusted endpoints. Type is not trust.

4. **Measurable residual.** CBCR localises residual integrity risk to false labelling and false endorsement, each weighted by reachable authority.
5. **Authority reduction and attenuation.** Untrusted-processing components hold no high-integrity capabilities, and delegation cannot amplify authority across a proof-carrying chain.
6. **Open propagation boundary.** CBCR does not solve black-box label propagation; it states the conservative rule and treats endorsement load as the key design tension.

19. Conclusion

Contract-Bound Cognitive Routing is a control architecture for agentic AI integrity. It does not attempt to make probabilistic reasoning deterministic. It **constrains** what can be constrained – control flow and closed structured data – by a verifiable reference monitor; it lets the model **reason** within a contained path that cannot reach a sink directly; and it **assures** the irreducible residual by confining it to one declared, fail-closed gate, bounding its reach, and measuring it.

The honest contribution is not that CBCR makes untrusted content safe. It does not. It is that CBCR draws a precise line between what is deterministically enforceable and what is not, reduces the untrusted-to-action attack surface beyond that line to declared, auditable, measurable endorsement boundaries, and bounds the reachable authority of what crosses them. A smaller claim than "the firewall stops malicious payloads" – and a stronger one. Constrain. Reason. Assure.

Appendix A — Minimal Contract Example

```
contract_id: agent.payments.support.v1
agent: payments_support_agent
status: approved
sources:
  - customer_ticket_text: untrusted
  - payment_status_api: admissible
routes:
  - customer_ticket_text → low_capability_reasoner
  - low_capability_reasoner → structured_ticket_summary
  - structured_ticket_summary → endorsement_gate.ticket_summary
sinks:
  - payment_status_lookup:
      required_integrity: admissible
      capability: read_payment_status
  - customer_email_reply:
      required_integrity: admissible
      capability: send_customer_message
forbidden:
  - raw_customer_text → payment_status_lookup
  - raw_customer_text → customer_email_reply
  - low_capability_reasoner → send_customer_message
endorsement_gates:
  - ticket_summary:
      methods:
        - schema_validation
        - pii_check
        - human_review_if_high_risk
logging:
  - route_decision
  - endorsement_event
  - sink_invocation
  - denial_event
```

Appendix B — Evaluation Checklist

Control question	Evidence required	Pass condition
Are all tool calls mediated by the firewall?	Route map, tests, logs	No unmediated path exists
Are all memory writes treated as sinks?	Memory architecture, policy config	Required integrity enforced
Are source labels assigned before reasoning?	Classifier config, test cases	All sources labelled
Does pinned control flow reject deviation?	Conformance tests	All deviations blocked
Does closed-value-space data reject malformed input?	Schema/validator tests	All structurally-invalid input rejected
Can untrusted model output reach tools directly?	Architecture test	No direct path exists
Are endorsement events logged?	Audit samples	Every promotion recorded
Is false-endorsement rate measured?	Adversarial test results	Rate reported by gate
Is reachable authority measured?	Capability register	Authority scope known
Does delegation attenuate capabilities?	Delegation-chain tests	Downstream subset verified
Is control-plane integrity protected?	Access control, change log	Only approved changes possible
Is label propagation conservative?	Output tests	Derived content remains untrusted until endorsed

Appendix C — Terminology

Term	Meaning
Untrusted	Low-integrity content that may contain adversarial instruction, error, or contamination
Triaged	Typed or classified by policy but not endorsed for high-integrity influence
Admissible	Endorsed for a specified route or sink under defined constraints
Trusted	High-integrity content or control-plane artefact accepted as authoritative under policy
Source	Any origin of content entering the graph
Sink	Any operation with real-world effect or trusted-context influence
Capability	Explicit authority to reach a route, tool, or sink
Contract	Policy object defining permitted capabilities and integrity requirements
Endorsement	Explicit operation that raises integrity status
Firewall	Reference monitor enforcing contracts, labels, and routing
Reachable authority	The set and severity of sinks reachable after endorsement
False endorsement	Erroneous promotion of malicious or unsafe content
Controllable core	The class of behaviour enforceable deterministically: conforming control flow and closed-value-space structured data

References

1. Anderson, J. P. (1972). *Computer Security Technology Planning Study*. ESD-TR-73-51, Vols. I & II, U.S. Air Force Electronic Systems Division, Hanscom Field, Bedford, MA (October 1972). – origin of the reference-monitor concept (complete mediation, tamper resistance, verifiability).
2. Biba, K. J. (1977). *Integrity Considerations for Secure Computer Systems*. MITRE Technical Report MTR-3153 (also ESD-TR-76-372), April 1977. – the integrity lattice and no-write-up / no-read-down discipline.
3. Denning, D. E., & Denning, P. J. (1977). Certification of programs for secure information flow. *Communications of the ACM*, 20(7), 504–513. <https://doi.org/10.1145/359636.359712> – information-flow control and label propagation.
4. Zdancewic, S., & Myers, A. C. (2001). Robust declassification. *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW-14)*, Cape Breton, Nova Scotia, 15–23. – the role of integrity in safe

declassification / endorsement.

5. Sabelfeld, A., & Sands, D. (2009). Declassification: Dimensions and principles. *Journal of Computer Security*, 17(5), 517–548. <https://doi.org/10.3233/JCS-2009-0352>
6. Willison, S. (2023). *The Dual LLM pattern for building AI assistants that can resist prompt injection*. simonwillison.net, 25 April 2023. – the quarantined/privileged construction CBCR adopts.
7. Debenedetti, E., et al. (2024). *AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents*. – the benchmark on which the adopted construction is evaluated.
8. Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., Kern, C., Shi, C., Terzis, A., & Tramèr, F. (2025). *Defeating Prompt Injections by Design*. arXiv:2503.18813. – CaMeL; the formalised dual-path construction.
9. The Linux Foundation (2026). *DNS-AID: Decentralized AI Agent Discovery* (announced 27 May 2026). github.com/dns-aid – DNS-based publishing, discovery, and verification of agents and MCP servers; the discovery/identity layer beneath CBCR.
10. Cloud Security Alliance AI Safety Initiative (2026). *MCP Security Crisis: Systemic Design Flaws in AI Agent Infrastructure*. CSA Labs, 4 May 2026. – industry recognition of MCP as a systemic agentic-AI attack surface; independently recommends mediating tool interactions through an explicit policy layer rather than model reasoning, the principle CBCR formalises.
11. Tan, et al. (2025). *ETDI: Mitigating Tool Squatting and Rug Pull Attacks in Model Context Protocol*. arXiv:2506.01333. – cryptographic, OAuth-attested tool-definition integrity; a layer complementary to CBCR's flow integrity, alongside DNS-AID's discovery and identity.

Verification note: all citations verified against primary sources (June 2026); foundational works 1–5 confirmed for title, venue, year, and pagination.

Citation, License, and Acknowledgements

© 2026 Rob Brennan. Released under [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/) – reuse and adaptation permitted with attribution. (Swap for "All rights reserved" to restrict reuse; CC BY is suggested because it legally compels the attribution that makes citation your return on this work.)

Cite as:

Brennan, R. (2026). *Contract-Bound Cognitive Routing: A Control Architecture for Agentic AI Integrity, Delegation, and Prompt-Injection Containment (v0.7)*. <https://doi.org/10.5281/zenodo.20590723> · <https://bounded.uk/cbcr>

```
@misc{brennan2026cbcr,  
  author      = {Brennan, Rob},  
  title       = {Contract-Bound Cognitive Routing: A Control Architecture for  
                Agentic AI Integrity, Delegation, and Prompt-Injection Containment},  
  year        = {2026},  
  note        = {Version 0.7},  
  howpublished = {\url{https://bounded.uk/cbcr}},  
  doi         = {10.5281/zenodo.20590723}  
}
```

Acknowledgements. This work was developed with the assistance of an AI system (Claude, by Anthropic) for technical review, structural editing, prior-art positioning, and figures. All claims, framing decisions, and the final text are the author's responsibility.