MULTIPLEWORKS

Implementation Pattern Annex

No. 01

AI That Knows Your Business

The Security Walkthrough — Where Responsibilities Sit Across One Request

For enterprise architects, senior risk, audit, and information security functions

MultipleWorks · Hong Kong · multipleworks.com.hk
Mark Goodchild · 2026

About this annex

This Implementation Pattern Annex is the third artefact in MultipleWorks' *AI That Knows Your Business* publication series. The Executive Briefing argues the strategic position to the executive sponsor (MultipleWorks 2026a). The Architecture Briefing specifies the architectural surfaces the enterprise architect designs against (MultipleWorks 2026b). This annex demonstrates how the security architecture the Architecture Briefing specifies operates in practice, by following one request through the four layers the Briefing's "Where responsibilities sit" diagram establishes. Each artefact in the series is published independently and updates on its own cadence; cross-references in this document point to specific sections in the Architecture Briefing that the annex elaborates.

The annex sits one altitude below the Architecture Briefing and one altitude above any specific runtime implementation. Where the Briefing names architectural surfaces and the responsibilities each surface carries, this annex shows the surfaces in operation against a single request, naming the public standards the architecture rests on and the points at which institutional discipline determines whether the architecture's assertions hold.

The audience is the same as the Architecture Briefing's primary audience: the enterprise architect designing AI integration into the institution's existing security and information management architecture. The annex is also written for the senior risk, audit, and information security functions who need to understand how the security property the Architecture Briefing asserts is actually produced. The regulator's analyst is the tertiary reader as before.

The IP boundary the Architecture Briefing establishes is restated and reinforced here. The architectural pattern is open and contributable. The standards the pattern relies on are public: RFC 8693 OAuth 2.0 Token Exchange, RFC 6749 OAuth 2.0, OpenID Connect, ISO/IEC 27701, NIST SP 800-207 Zero Trust Architecture, and the data-architectural patterns established in the Architecture Briefing's section 4. Score is the open behaviour governance specification operating above runtime formats. Score-compliant runtimes are commercial implementations the institution selects; this annex describes the runtime category and what runtimes do at the architectural pattern level. The annex does not describe the internals of any specific runtime, including MultipleWorks Maestro. That boundary is structural rather than rhetorical.

The annex is published under the same Creative Commons BY-ND 4.0 terms as the Architecture Briefing. The suggested citation is:

> *MultipleWorks (2026c). AI That Knows Your Business: Implementation Pattern Annex — The Security Walkthrough. MultipleWorks Limited, Hong Kong. Available at multipleworks.com.hk/briefings.*

About the author

Mark Goodchild is Founder and Managing Director of MultipleWorks, a Hong Kong consultancy specialising in enterprise AI architecture and governance for regulated APAC enterprises. His background spans 25 years of enterprise architecture and digital transformation, including eleven years at EY, leaving as a Director in the APAC emerging tech consulting practice.

Table of contents

CHAPTER 01

One request, four layers

A relationship manager at an APAC private bank opens the institutional AI assistant and asks a question. The question is concrete: summarise the recent portfolio activity for one of her clients and flag any transactions worth a closer look. The principal is authenticated, her client book is on the institution's CRM, and the AI has access to the institution's transaction systems through governed tool definitions. By the time the AI returns its answer, every architectural layer the Briefing specifies has done a piece of identifiable work. The walkthrough that follows takes that single request and traces it through each layer in turn.

The relationship manager does not see most of what happens. She sees a question and an answer. The architecture is calibrated specifically to keep it that way for her, while producing the audit evidence the institution needs to demonstrate to a regulator that nothing in the request escaped institutional control. Two readers therefore matter at this altitude: the enterprise architect, who needs to know what each layer is doing and what evidence each layer produces, and the audit and compliance reader, who needs to know that the evidence is sufficient for the institutional accountability framework to operate on.

Four layers are involved. The first is the institutional infrastructure where identity and access are decided. The second is the AI gateway, where the institutional principal's credentials are exchanged for the scoped tokens the AI will operate under. The third is the Score-compliant runtime, where the institutional behaviour specification is enforced against the request and the data context is filtered before the model sees anything. The fourth is the model substrate, where the language model produces the response. The Briefing's diagram fixes the order and the responsibilities. This annex unpacks them.

Figure 1. Where responsibilities sit across the architecture. The LLM is downstream of every access decision; it does not make them.

Two patterns recur across the layers and matter for the security property the architecture produces. The first is the principal-actor distinction: the human principal authenticates, the AI service is the actor operating under the principal's authority for the duration of the request, and audit captures both identities at every layer. The second is the supports-but-does-not-constitute axiom: each layer carries a piece of the architectural pattern, none of them carries the whole, and the institutional accountability framework operates on the evidence the layers produce together. The architecture is the artefact that supports institutional accountability; it does not replace it.

The architecture works in this case because the layers were designed together. None of them works in isolation.

CHAPTER 02

Layer 1, institutional identity decides

The first decision in the architecture is institutional and predates the AI estate by years. The relationship manager logs into the institution's identity provider, authenticates against the institutional credential set, and receives the institutional access claims her role and her assignments produce. The identity provider is Microsoft Entra ID, Okta, AWS IAM Identity Center, or whatever institutional system the institution already operates; the AI estate inherits this layer rather than reinventing it. The Briefing's section 3 treats this layer as the institutional spine the AI estate is grafted onto. The integration is what is new; the identity infrastructure is not.

What the identity provider decides at this layer is what the relationship manager is permitted to read, write, and invoke against the institutional infrastructure. The decision is human-principal-shaped and predates any AI consideration. The relationship manager has access to her client book, has read access to the transaction systems for those clients, has the ability to invoke certain tools (compliance flagging, internal escalation), and has no access to other relationship managers' clients or to records held by other lines of business. These access claims are the institutional baseline. They follow the relationship manager through every system she touches, including the AI estate.

The AI is not present in this decision. The model has no role at Layer 1, the runtime has no role at Layer 1, the gateway has no role at Layer 1. The institutional access framework decides what the relationship manager is authorised to do, and the AI estate inherits the decision. This is the architectural inversion the Briefing's "Where responsibilities sit" diagram makes explicit and the architectural error caption names. Treating the LLM as the access decision point produces unsound architectures because the LLM is downstream of every decision and cannot enforce what it cannot see. Layer 1 is where the decision actually sits, and the rest of the architecture is calibrated to carry the decision through to the model substrate without losing it.

What the layer produces, beyond the access decision itself, is the authentication evidence the rest of the architecture operates on. The session is authenticated, the principal claim is signed, and the institution's audit infrastructure captures the authentication event as it would for any other system the relationship manager touches. The AI gateway in chapter 3 will consume this evidence. The runtime in chapter 4 will operate against it. The audit record at the end of the request will include it. The architecture does not invent identity for the AI estate; it inherits identity from the institution and carries it forward.

The institutional discipline at this layer is the conventional discipline. Where the institution's identity architecture is sound, the AI estate is sound here by inheritance. Where the institution's identity architecture has gaps (over-broad role assignments, stale entitlements, weak break-glass procedures), the AI estate inherits the gaps. The Briefing names this honestly. The work the institution does to govern the AI estate at this layer is the work the institution should already be doing for everything else.

CHAPTER 03

Layer 2, the gateway brokers credentials

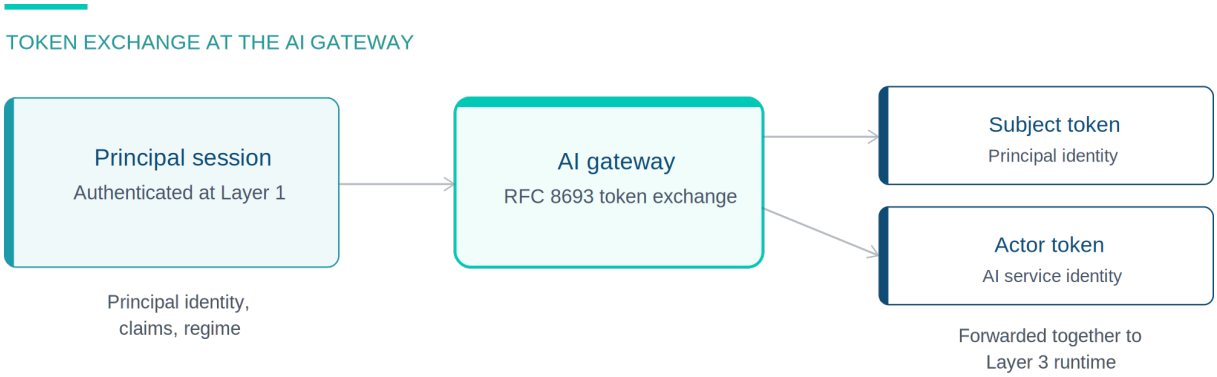

Figure 2. Token exchange at the AI gateway. The AI never operates under the principal's full institutional credentials.

The AI gateway is where the institutional principal's authenticated session becomes the request the AI estate operates on, and the architecture stands or falls on the integrity of the exchange. The relationship manager's session, authenticated at Layer 1, presents at the gateway as the inbound request the AI estate has to act on. The gateway's job at this layer is to convert that authenticated session into per-request, scoped credentials the runtime can use to mediate retrieval and tool invocation downstream. This is RFC 8693 OAuth 2.0 Token Exchange operating at the architectural element the Briefing's section 5 elevates to a layer of its own.

Two tokens emerge from the exchange. The subject token represents the relationship manager: her institutional identity, her access claims, the regulatory regime under which her session operates, the authentication strength the institutional identity provider produced. The actor token represents the AI service: a separate institutional identity issued to the AI estate, scoped to the specific request, time-bounded to the request's lifetime, and tied to the subject token through the exchange. The pair carries the principal-actor distinction the Briefing's section 3 establishes through the rest of the architecture. The audit infrastructure at section 7 captures both. The institutional accountability framework operates on both.

The credentials issued by the gateway are deliberately narrower than the relationship manager's broader institutional access. Where the relationship manager has read access to her entire client book at the institutional CRM, the request-scoped credentials issued at the gateway might restrict the AI's access to the specific client her query named, or to a clearly-bounded subset of her book the request engages. Where the relationship manager has tool access to the compliance escalation system, the per-request credentials might withhold

tool access entirely if the request is informational, and grant tool access only when the request explicitly invokes a tool the institution has decided the AI may invoke on the relationship manager's behalf. The narrowing is done at the gateway, against the institutional behaviour specification the runtime will execute, before the credentials reach the runtime. Defence in depth begins here.

The architectural property this layer produces is that the AI never operates under the relationship manager's full institutional credentials. The AI operates under credentials that are a subset of hers, scoped to the request, time-bounded, and revocable. If the gateway revokes the actor token mid-request (because the request has gone too long, because a downstream compliance signal has fired, because the institutional behaviour specification requires it), the runtime cannot continue the request against expired credentials. The token has to be re-exchanged or the request has to fail closed. This is the standard zero-trust pattern from NIST SP 800-207 applied at the AI gateway specifically, and the architectural reason the gateway is a layer of its own rather than a feature of one of the systems on either side of it.

The gateway holds trust relationships with downstream systems. The AI does not. When the runtime calls the institutional CRM in chapter 4, it presents the scoped actor token; the CRM trusts the gateway, validates the token, and applies its own access controls against the principal claims the token carries. The runtime is mediating, not authenticating. The CRM's access decision is the CRM's, made against its own framework, with the gateway-issued token as the input. The architecture does not require the AI estate to inherit trust from systems that did not grant it.

The most common implementation error at this layer is to short-circuit the exchange. A gateway that accepts the relationship manager's session token and forwards it to downstream systems unchanged collapses the principal-actor distinction and breaks the audit record. The runtime appears in audit logs as the relationship manager rather than as itself, and forensic reconstruction after a security event becomes substantially harder. RFC 8693 exists precisely to prevent this, and the institutional pattern at the gateway is to make the exchange non-optional.

CHAPTER 04

Layer 3, the runtime mediates

The Score-compliant runtime is the architectural element where the institutional behaviour specification becomes operational against the request. The relationship manager's question has reached the runtime now, accompanied by the gateway-issued scoped credentials and the audit context the gateway produced. The runtime's job is to execute the institutional behaviour specification against this request, mediate the data retrieval and tool invocation the request requires, and produce the filtered context the model substrate will eventually see. The Briefing's section 9 specifies the six runtime responsibilities. This chapter walks the security-relevant subset.

The first runtime act is to compile the institutional behaviour specification into the form the model endpoint requires for this request. The specification is authored once at Score's level of abstraction; the runtime compiles it to the vendor-native format at request time. For a model accessed through the Anthropic API, the compilation produces tool definitions and a system prompt the API expects. For a model accessed through the Microsoft Agent Framework, the compilation produces the equivalent in MAF's form. For an on-premise model accessed through a vendor-neutral inference endpoint, the compilation produces whatever the endpoint accepts. The institution authors at one altitude; the runtime executes at the altitude each model substrate consumes. The architectural argument the Briefing's section 8 makes is that this compilation step is structurally necessary, and the runtime is the architectural element that performs it.

The second runtime act is to mediate retrieval against the scoped credentials. The relationship manager's query mentions a specific client. Retrieval in 2026 is typically a vector search against an embedding store of the institution's transaction history, augmented with a structured query against the institutional CRM and transaction systems. The runtime issues these queries with the gateway-scoped credentials, and the downstream systems apply their own access controls to the queries against the principal claims the token carries. Records the relationship manager has no right to read are not returned by the downstream systems. This is the institutional access framework operating in its conventional form.

The retrieval is where the architectural pattern starts doing security work that conventional systems do not. The embedding store will return chunks that match the query semantically, regardless of access intent. A vector match for "Mr Wong recent activity" might surface chunks from an internal compliance investigation the relationship manager has no right to see, chunks from another client whose transactions look structurally similar, or chunks from a regulatory filing that is public to all staff. The institutional access framework operates against records, not against semantic chunks. The runtime has to bridge the gap. The pattern that bridges it is the two-stage enforcement that chapter 5 treats in its own right.

The third runtime act is to mediate tool invocation. When the model decides to call a tool (to flag a transaction for compliance review, to escalate to a human reviewer, to log the interaction in the institutional CRM), the runtime intercepts the tool call before it executes. The tool's definition was compiled into the request from the institutional behaviour specification at the start of the request, with whatever scoping the specification carried. The runtime checks the tool call against the scoped credentials, against the institutional behaviour specification, and against any policy gates the specification carried (rate limits, dual-control requirements, approval thresholds). Tool calls that fail the check are rejected without executing; the runtime records the rejection in the audit infrastructure and either returns the rejection to the model for graceful degradation or fails the request closed, depending on how the institutional specification was authored.

The fourth runtime act is to produce the audit evidence the institutional accountability framework will operate on. The runtime captures the principal claim, the actor claim, the specification version in effect at request time, the regulatory mappings the specification carried, the records retrieved, the records denied, the tools attempted, the tools mediated, the tools executed, and the model endpoint that produced the response. The audit infrastructure at section 7 schemas this evidence at the AI gateway. The institutional accountability framework operates on the captured schema.

What the runtime does not do at this layer is decide policy. The institutional behaviour specification is what decides policy; the runtime executes the specification. If the specification permits a tool call, the runtime mediates the call. If the specification forbids it, the runtime rejects it. The runtime is not a policy engine that makes its own decisions; it is the architectural element that operationalises the specification at request time. This is the supports-but-does-not-constitute axiom in concrete form, and it is the structural reason the runtime category is portable across institutional environments.

CHAPTER 05

Pre and post-retrieval enforcement at the runtime

TWO-STAGE ENFORCEMENT AT THE RUNTIME LAYER

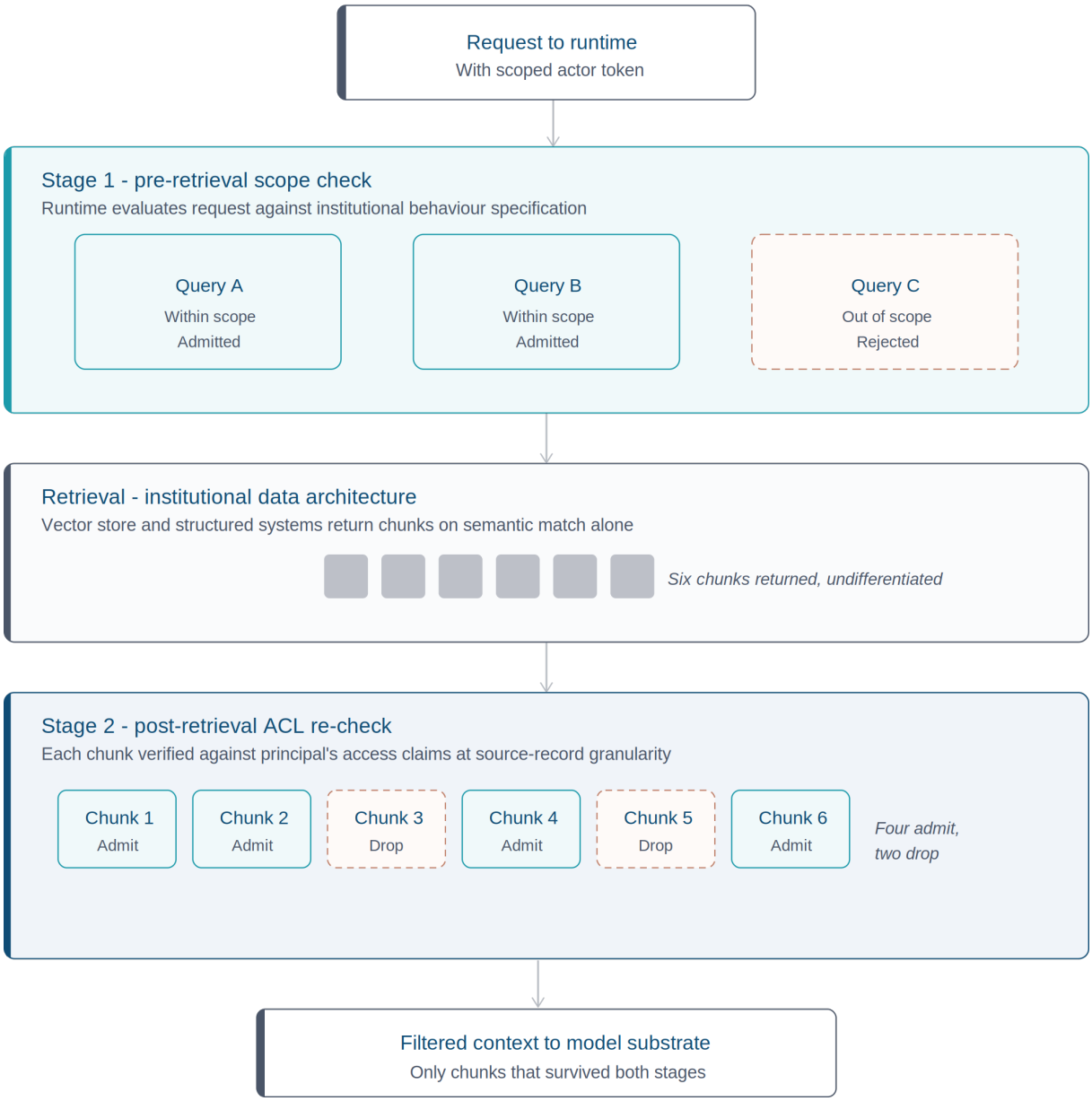

Figure 3. Two-stage enforcement at the runtime layer. Vector match returns chunks; institutional access decides which ones the model sees.

The Briefing's section 3 establishes two stages of enforcement against data leakage at the architectural altitude: per-principal scope filtering (Stage 1) and per-classification LLM routing (Stage 2). This chapter elaborates the operational detail of how the runtime delivers the Briefing's Stage 1: the work that ensures the model substrate sees only data the principal is authorised to see. The Briefing's Stage 2, the routing of confidential workloads to the on-premise LLM tier and non-confidential workloads to the hyperscaler tier, is treated at chapter 6 in the context of which model substrate sees the request at all.

Within the Briefing's Stage 1, the runtime performs two distinct checks: a pre-retrieval scope check against intent, and a post-retrieval ACL re-check against the data the retrieval surfaced. The architectural property the Briefing asserts at the runtime layer is that, if the principal cannot read a record, no retrieval against that record reaches the model substrate. The property holds only if both checks happen, and the most common failure mode in 2026 implementations is to perform only one of them.

The first check is pre-retrieval scope enforcement. The runtime, before issuing any retrieval query against the institutional data architecture, evaluates the request against the institutional behaviour specification and the scoped credentials. The specification states what categories of data the AI is permitted to retrieve for this principal in this context: transaction records for the relationship manager's own client book, yes; transaction records for other relationship managers' clients, no; internal compliance investigations, no; regulatory filings public to staff, yes. The runtime issues only the queries the scope permits. Queries outside scope are rejected at the runtime layer and never reach the institutional data systems.

The second check is post-retrieval ACL re-check. Of the queries that pass the scope check and execute against the institutional data architecture, the records returned are checked again against the principal's access claims before any of them is admitted to the model context. This re-check is necessary because retrieval against a vector store is not access-controlled in the conventional sense. The vector match returns chunks that are semantically close to the query, regardless of whether the principal has read access to the source records the chunks were derived from. A 2024 academic literature on retrieval-augmented generation identifies this leak path as one of the most consequential security risks for enterprise RAG. The architectural answer is the post-retrieval re-check.

The two checks do different work. The pre-retrieval check prevents the runtime from formulating queries the principal has no right to issue. The post-retrieval check prevents the runtime from injecting retrieved chunks the principal has no right to read into the model context. The pre-retrieval check operates on the request's intent. The post-retrieval check operates on the data the request actually surfaced. Implementing only pre-retrieval leaves the vector-leak path open. Implementing only post-retrieval issues queries beyond the principal's scope and creates audit-noise without delivering the outcome. Both are required.

The pattern is well-established in the broader security literature. Pre-action authorisation and post-action re-validation is the zero-trust pattern from NIST SP 800-207 applied at every architectural boundary. The novelty in the AI estate is that the post-action re-validation has to handle vector matches that the conventional access framework was not

designed to handle. Some institutional vector stores in 2026 are starting to expose ACL-aware retrieval natively: the AWS Bedrock Knowledge Bases pattern, where deletion at the source bucket cascades through the embedding store, is an example of the lineage discipline this chapter is the security counterpart of. Where the institutional vector store is not ACL-aware natively, the runtime has to perform the re-check explicitly: every retrieved chunk is associated with its source record's lineage; every source record's classification and access controls are applied against the principal's claims; chunks failing the re-check are dropped silently before the model context is assembled.

The design-time decision the institution makes is whether to require ACL-aware retrieval as a procurement criterion for vector stores admitted to the institutional data architecture, or to require runtimes to implement post-retrieval re-checking explicitly against vector stores that do not expose the capability natively. Both are sound architectural answers. The architectural failure is to assume the vector store's index and the institutional access framework agree without explicit enforcement. They do not.

The relationship manager's request, by the time it has passed both checks, carries only the records she is institutionally authorised to read, retrieved through queries she was permitted to issue, with the audit evidence that demonstrates both. The model substrate in chapter 6 will see only this filtered context. The architectural assertion the Briefing makes is operational at this point and not before.

CHAPTER 06

Layer 4, what the model substrate sees

The model substrate is the foundation model the request eventually reaches: an Anthropic Claude endpoint, a GPT model in the Microsoft 365 stack, an Alibaba Qwen instance, a Google Gemini endpoint, an on-premise open-weights model, or whatever combination the institution's multi-vendor architecture selected for this class of request. The model substrate is, architecturally, untrusted in the access-decision sense. It does not authenticate the principal, does not decide what data it sees, does not enforce institutional policy, does not have its own access framework operating in parallel to the institution's. The Briefing's "Where responsibilities sit" diagram caption names the architectural error of treating the LLM as the access decision point; this chapter is the positive form of the same statement.

What the model substrate receives is a fully-prepared context: the system prompt compiled from the institutional behaviour specification, the principal's question, the retrieved chunks that survived chapter 5's pre and post-retrieval enforcement, the tool definitions the institutional specification permits for this request, and whatever conversation history is in scope for this principal-actor pair. The context is the model's view of the world for the duration of this request. Everything outside the context is invisible. Other principals' data is invisible because the runtime did not retrieve it. Records the principal cannot read are invisible because they failed the post-retrieval re-check. Tools outside the request's permitted scope are invisible because the institutional specification did not include them. Cross-tenant context is invisible because the architecture does not produce it.

Which model substrate the request reaches is itself an enforcement decision. The Briefing's section 3 names this as Stage 2, per-classification LLM routing, and the runtime is the architectural element that delivers it at request time. The runtime evaluates the classification of the request (the classification of the data the principal is operating against, the classification implied by the principal's role, the classification carried in the institutional specification) and routes the request to the model endpoint policy permits for that classification. Confidential workloads route to the on-premise LLM tier. Non-confidential workloads route to the hyperscaler or APAC vendor tier. The two tiers do not share endpoints. A request mis-routed to a tier its classification forbids is rejected at the runtime layer; the rejection is a security event captured in audit. The relationship manager's request in this annex's running example carries the classification of the client portfolio data the request retrieves, and the runtime routes the request to the LLM tier the institution has approved for that classification. The Briefing's section 5 deployment topology is what makes the tier separation operational; the runtime is what enforces the routing at request time.

The model produces a response against this filtered context. The response is in the model's native register, with whatever capability the model's training and the prompt design allow. The runtime in chapter 4 is what mediates the response: tool calls the model emits are intercepted; the model's claims about what it retrieved are reconciled with what the runtime actually injected; the response that returns to the relationship manager has been through the runtime once more on its way out. The model is not the last layer the response touches.

The architectural property this produces is that the model is downstream of every access decision in the architecture. If the relationship manager could not read a record at Layer 1, the runtime did not retrieve it at Layer 3, and the model never had it at Layer 4. The model cannot exfiltrate what it never received. The model cannot be jailbroken into revealing what it cannot see. The most ambitious prompt-injection attack against a model substrate cannot recover data the runtime never allowed into the context. The architectural defence holds because the data was filtered before generation began, not because the model resisted disclosure during generation. This is the structural reason the four-layer pattern is preferred over the prompt-shielding patterns common in 2024 implementations.

The institution's procurement choice between model substrates is a substantive question and the Briefing's section 14 treats it. From the security architecture's perspective, the choice is constrained by what the runtime supports rather than by the model's own capabilities: a model the runtime cannot mediate effectively is a model the institution cannot deploy, regardless of the model's substantive quality. The architecture pushes the institution toward model substrates the runtime can wrap completely. Models exposed only through APIs that bypass the runtime are not deployable under this pattern; the institutional discipline is to procure models through endpoints the runtime can mediate.

The model is the most discussed layer of the architecture and the layer that does the least work in the security architecture. This is by design. The architecture supports institutional accountability without constituting it; the model is the architectural element where institutional accountability is least relevant because the institution already constructed the conditions under which the model could operate safely.

CHAPTER 07

The hard cases the pattern has to handle

Five cases stress the architecture beyond the linear walkthrough so far. Each is real, each appears in production deployments, and each is where insufficient architectural patterns fail. The institutional discipline is to confront each case at design time and to verify that the runtime category the institution selects handles each case explicitly.

The first case is the model-initiated tool call the runtime has not seen before in this request. The relationship manager asked a question, the model decided that flagging an unusual transaction was the right response, and the model emits a tool call to the institutional compliance system. The runtime mediates the tool call against the scoped credentials and the institutional behaviour specification, as chapter 4 establishes. The hard variant is when the tool call's parameters include data the model surfaced from an earlier turn of the conversation, or that the model inferred from its training rather than retrieved from the institutional data architecture. The runtime cannot blindly trust model-supplied parameters. The pattern is to validate the parameters against the institutional specification before executing the tool call, treating the model's parameters as untrusted input that has to clear the same scoping the original retrieval cleared. Tool calls that include parameters outside the request's scope are rejected even if the tool itself is permitted.

The second case is the streaming response that begins generating before retrieval is complete. Some model endpoints stream tokens as they generate; some institutional applications expect streaming responses to keep the relationship manager's interface responsive. The architectural question is whether streaming changes the security model, and the answer is that it should not. The runtime's two-stage enforcement happens before generation begins, on the context the model is about to receive. Streaming is the form the response takes back to the principal, not a parallel data path that bypasses the runtime. The implementation discipline is that no token of generated output is released to the principal until the runtime has accepted the model's outputs against the institutional specification, which can be done at sentence or chunk granularity rather than at full-response granularity. The streaming gain is interface latency; the security gain at the architectural pattern level is unchanged.

The third case is the multi-turn conversation where the principal's credentials at turn three are not the credentials that authorised turn one. Sessions expire. Roles change. Compliance signals fire mid-conversation. The architectural pattern is that every turn re-validates against the current credentials at the gateway: the actor token is re-issued or refreshed at each turn, the runtime accepts only the current token, and the conversation history is itself accessed under the current principal's scope rather than under the credentials that wrote earlier turns. Conversation history that includes records the current principal can

no longer access is filtered out of the context for the new turn. This is uncomfortable for product designers, who would prefer the conversation feel continuous. The architectural answer is that institutional security is the property the architecture produces, and the conversation's continuity is a presentation choice that has to live within the security envelope rather than beneath it.

The fourth case is right-to-erasure cascading through the four-layer data taxonomy from the Briefing's section 4. When a record is erased at the source data layer, the architectural requirement is that derivative artefacts at every downstream layer are erased in turn: the embeddings derived from the source data, the conversation history that references them, the inference logs that captured them, and any fine-tuning corpus the records contributed to. The Briefing's section 11 treats the lifecycle architecture that produces this property at the data layer. The security-architectural complement is that the runtime cannot retrieve embeddings whose source records have been erased, even if the embedding store has not yet completed the cascade. The institutional discipline at the runtime layer is to enforce a freshness check against source-record lineage at retrieval time, not to assume the embedding store and the source data architecture are synchronised. The discipline is uncomfortable because it adds latency at retrieval. The architectural answer is that erasure is a structural property of the institutional data architecture, and the runtime carries the property forward rather than trusting it.

The fifth case is the cross-jurisdictional request, where the principal authenticates in Hong Kong, the data the request retrieves resides in Singapore, the model substrate is hosted in Tokyo, and the regulatory mesh from the Briefing's section 13 specifies overlapping obligations from HKMA, MAS, the EU AI Act for any principal in the European Economic Area on the call, and the mainland Chinese stack if any retrieval reaches into the mainland. The architectural pattern requires that the institutional behaviour specification carries the jurisdictional mapping the request triggers, that the runtime selects model endpoints whose physical residency is consistent with the data residency and the regulatory obligation, and that the audit evidence captures the jurisdictional decision the request made. This is not theoretical. The relationship manager at the APAC private bank in this annex's running example exists in this jurisdictional configuration regularly. The architecture's contribution is to make the jurisdictional decision explicit at request time rather than discovered during a regulatory inspection.

The hard cases are not edge conditions. They are the steady state of operating an AI estate at institutional scale. The architecture is calibrated for them; the institutional discipline is to confirm the calibration at design time.

CHAPTER 08

What the runtime does not do

The architectural pattern is at least as strong in what it withholds from the runtime as in what it grants. The runtime is the architectural element where most of the new work in the AI estate appears, and the institutional temptation is to treat the runtime as the place where every problem the AI estate raises can be solved. The temptation produces unsound architectures. Several functions the runtime might appear to be a candidate for are deliberately held elsewhere, and naming the negative space is part of the discipline the pattern requires.

- **The runtime does not authenticate the principal.** Authentication happens at the institutional identity provider at Layer 1, and the AI gateway at Layer 2 consumes the authenticated session. A runtime that performs its own authentication is operating outside the architectural pattern, and the institutional accountability framework will struggle to operate against the principal-claim divergence such a runtime produces.
- **The runtime does not issue credentials.** The gateway issues credentials through the OAuth 2.0 Token Exchange pattern at chapter 3. A runtime that issues its own credentials breaks the principal-actor distinction the audit infrastructure relies on, and the runtime appears in audit logs as having acted on its own authority rather than as the actor for an institutional principal. Forensic reconstruction after a security event is substantially harder where this discipline has been compromised.
- **The runtime does not own the institutional data classification.** The classification framework operates at the institutional data architecture from the Briefing's section 4. A runtime that re-derives classification from the data it sees at retrieval time will diverge from the institutional truth and produce the inconsistency between the AI estate and the rest of the institution that the Briefing's regulatory mesh treatment specifically warns against.
- **The runtime does not store credentials beyond the request lifetime.** The architectural pattern is that credentials are scoped to the request, time-bounded, and revocable, as chapter 3 establishes. A runtime that caches credentials for performance reasons, even with the best intentions, undermines the gateway's revocation capability and produces a class of audit gaps that regulators have begun to identify in 2026 as material control weaknesses.
- **The runtime does not modify or transform authorised data without an explicit specification authorisation.** The runtime mediates retrieval and tool invocation; it does not silently reshape the data the institution authorised. Where the institutional specification calls for data transformation, the transformation is named in the specification and audited as such. Silent transformation is the path by which institutional data integrity erodes.

- **The runtime does not make policy decisions outside the institutional specification.** The policy framework is the institution's, and the institutional behaviour specification is its operational expression. A runtime that develops its own policy logic, however well-meaning, undermines the institutional accountability framework's primacy. The runtime is the architectural element that operationalises the specification; the specification is what makes policy decisions.
- **The runtime does not bypass the institutional access framework.** Where the runtime's view of the institutional access framework and the institutional access framework's own state diverge, the institution's framework wins. The runtime is downstream of the access framework, not parallel to it. A runtime that develops its own access-decision logic for performance reasons or for closing latency gaps is producing the second-system problem at the institutional access layer.
- **The runtime does not retain conversation context outside the lifecycle architecture.** The Briefing's section 11 treats the lifecycle architecture for conversation history and inference logs. The runtime contributes to lifecycle architecture as a producer of the conversation history layer; it does not develop its own retention logic outside the institutional lifecycle.

The negative space matters because the architectural pattern's portability across compliant runtimes depends on what the runtimes share, which is largely what they refrain from doing. Runtimes that respect the negative space are interchangeable from the institution's perspective. Runtimes that absorb functions held elsewhere become the institution's only choice, and the portability claim that motivates the architectural pattern weakens correspondingly.

CHAPTER 09

The audit evidence the architecture produces

The architecture's contribution to institutional accountability is the audit evidence the layers produce together. The Briefing's section 7 schemas the evidence at the AI gateway and the layers above and below the gateway populate the schema. The institutional accountability framework operates on the schema. Regulators inspect the schema. Forensic reconstruction after a security event uses the schema as its primary substrate. The schema is what the architecture produces, and the institutional value of the architecture is largely the schema's quality.

For the relationship manager's request in this annex's worked example, the schema captures a structured record of the entire request lifecycle. The principal claim is the relationship manager's institutional identity, the strength of authentication the institutional identity provider produced, the role and entitlements active at the moment of authentication, and the regulatory regime under which the principal's session operated. The actor claim is the AI service identity issued to the AI estate for this request, the scope under which the actor token was issued, the time-bound the gateway placed on the token, and the lineage to the subject token through the OAuth 2.0 Token Exchange.

The specification version is the institutional behaviour specification active at request time, identified by version, attributable to the institutional editor or committee that approved the version, and tied to the regulatory mappings the version carried. The regulatory mappings name the specific provisions of EU AI Act, NIST AI RMF, ISO/IEC 42001, HKMA expectations, and the mainland Chinese stack the specification served at the moment of the request. Where the request triggered a specific regulatory obligation, the trigger is captured.

The retrieval record names the queries the runtime issued, the queries the institutional data architecture executed, the records the queries returned, the records that survived the post-retrieval re-check, and the records that did not. Records that did not survive the re-check are captured by reference to their access-control reason rather than by their content; the audit evidence demonstrates that the re-check fired without itself becoming a vehicle for unauthorised disclosure. Lineage references to the source records, the embedding store, and the retrieval system are captured at the granularity the institutional data architecture maintains.

The tool record names the tool calls the model emitted, the tool calls the runtime mediated, the tool calls that executed, the tool calls that were rejected, and the basis for each rejection. Where tool calls executed against external institutional systems, the audit at the gateway is the AI estate's record; the institutional system's own audit is the corresponding record on the receiving side, and the two are reconcilable through the request identifier the gateway propagates. Forensic reconstruction across the AI estate and the institutional systems it touched operates on this reconciliation.

The model record names the model endpoint that produced the response, the version of the model active at the moment, the parameters the runtime supplied, and the response the model produced. The model response is captured with the runtime's output mediation applied; raw model outputs, where they differ from the mediated outputs, are captured separately for the cases where the institutional accountability framework needs to inspect the difference.

The schema is structured for both regulatory inspection and operational use. The regulatory inspection use case is the supervisory examination that asks specific questions about specific requests. The operational use case is the institutional audit and risk management discipline that operates on the schema continuously, looking for patterns that warrant attention before they become incidents. The institutional audit function from the Briefing's audience is the consumer of the operational use; the regulator's analyst is the consumer of the supervisory use. The schema is the same for both.

What the schema does not contain is interpretation. The schema captures what happened. The interpretation (whether the institution discharged its accountability properly, whether the regulatory obligations were met, whether the institutional behaviour specification was sufficient for the case at hand) is the work of the institutional accountability framework operating on the schema. The architecture supports the framework. The framework discharges the accountability.

The architecture's contribution to the institution's regulatory posture is to make the conversation between the institution and the regulator possible at the granularity 2026 supervision expects. The institution that has built the architecture has the evidence the regulator inspects. The institution that has not built the architecture has to assemble the evidence reactively, from systems that were never designed to produce it, in the time the supervisory examination allows. The architecture is the difference between a regulatory conversation the institution can have and one the institution cannot.

CHAPTER 10

Closing

The security walkthrough this annex presents is one request through four layers, traced at the architectural pattern altitude. The institutional architect reading the Architecture Briefing now has the implementation pattern the Briefing's section 9 specifies in concrete enough form to brief the institutional security, audit, and identity teams whose existing architectures the AI estate is being grafted into.

The companion artefacts in the published *AI That Knows Your Business* series serve their own readers. The Executive Briefing argues the strategic position to the executive sponsor. The Architecture Briefing specifies the architectural surfaces to the enterprise architect. This Implementation Pattern Annex demonstrates the security architecture to the practitioners who design and operate it. Each operates at its own altitude. The series together is the institutional conversation the architecture supports.

The architecture supports institutional accountability. The institution discharges institutional accountability. The architecture is the artefact through which the discharge is made possible. The work belongs to the institution.

Mark Goodchild *is Founder and Managing Director of MultipleWorks, a Hong Kong consultancy specialising in enterprise AI architecture and governance for regulated APAC enterprises. His background spans 25 years of enterprise architecture and digital transformation, including eleven years at EY, leaving as a Director in the APAC emerging tech consulting practice. Reachable at hello@multipleworks.com.hk.*

MultipleWorks (2026c). AI That Knows Your Business: Implementation Pattern Annex — The Security Walkthrough. MultipleWorks Limited, Hong Kong. multipleworks.com.hk/briefings

v1.1 · May 2026

References

This Annex draws on the standards and primary sources catalogued in the Architecture Briefing's references. This section names the specific standards the Annex's security walkthrough rests on most directly.

Primary standards

IETF. 2019. *RFC 8693: OAuth 2.0 Token Exchange.* Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc8693

IETF. 2012. *RFC 6749: The OAuth 2.0 Authorization Framework.* Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc6749

NIST. 2020. *SP 800-207: Zero Trust Architecture.* National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

ISO/IEC. 2019. *ISO/IEC 27701: Privacy Information Management.* International Organisation for Standardisation.

NIST. 2022. *SP 800-92 Rev. 1: Guide to Computer Security Log Management.* National Institute of Standards and Technology.

Companion publications

MultipleWorks. 2026a. *AI That Knows Your Business: Executive Briefing.* Hong Kong: MultipleWorks Limited. https://multipleworks.com.hk/briefings

MultipleWorks. 2026b. *AI That Knows Your Business: Architecture Briefing.* Hong Kong: MultipleWorks Limited. https://multipleworks.com.hk/briefings

MultipleWorks. 2026c. *AI That Knows Your Business: Implementation Pattern Annex — The Security Walkthrough.* Hong Kong: MultipleWorks Limited. https://multipleworks.com.hk/briefings (this document).