loopthink

THE SOVEREIGN AI CONTROL PLANE

TECHNICAL WHITEPAPER

Governing MCP Access to Enterprise Systems

Gateway architecture, field-level masking, and deployment topologies — for architects, security engineers and CISOs evaluating this class of product.

ABSTRACT

The Model Context Protocol has made it trivial to connect a general-purpose AI assistant to an internal system. A working connection to an ERP or a production database is now an afternoon of work rather than a project. That convenience has outrun the controls around it: in most organizations, nobody can answer how many such connections currently exist, which identities they run under, or what data has crossed the boundary.

This paper describes what a governing layer between AI clients and internal systems has to do, how field-level masking works and where its limits are, and how three deployment topologies differ in the concrete question that matters to a security architect: what actually leaves the network, and who holds the credentials.

It is written for architects, security engineers and CISOs evaluating this class of product. It is deliberately explicit about what a gateway does *not* solve.

1 Why the control point moved

Before MCP, integrating an AI assistant with an internal system meant building a bespoke service: authentication, API wrapper, error handling, deployment. The cost of that work was itself a control. Projects went through architecture review because they had to go through engineering.

MCP removes that cost. A developer can expose an internal database as a set of callable tools in an afternoon, and any MCP-capable client — ChatGPT, Claude, Copilot, Cursor, a self-hosted UI — can consume it. The protocol is now governed under the Agentic AI Foundation with the major model vendors and several enterprise software vendors participating, so this is not a bet on a single vendor's roadmap. It is converging toward default infrastructure.

The consequence for security architecture is specific. **Authorization has moved from the integration to the session.** In the old model, a service account had a fixed scope, defined at build time and reviewed once. In the MCP model, a human identity connects a client to a server and then issues open-ended natural-language requests against whatever tools that server exposes. What the user can do is determined at connection time, per user, and changes as their entitlements change.

If nothing sits between the client and the system, three things are true simultaneously:

- The internal system sees a valid credential and cannot distinguish a sanctioned agent from a developer's laptop or a personal ChatGPT account.
- Tool-call detail — which tool, which parameters, which identity, which system, at what time — is not captured anywhere in the format an audit or incident investigation requires. Conversation logs from the model vendor do not contain it.
- Payload data, including personal data, transits to the model provider in whatever form the tool returned it.

A gateway exists to make each of those three false.

2 What a governing gateway has to do

Five functions, in the order a request encounters them.

Identity binding. The MCP session must be tied to a real, named user from the organization's existing directory, not to a shared API key. Practically this means the gateway acts as an OAuth resource server and authenticates the session against the customer's identity provider. Every downstream action inherits that identity.

Tool curation and scoping. Not every user should see every tool. The gateway resolves the authenticated identity against a role model and returns only the tools that identity is permitted to call. A finance controller and a support agent connecting to the same endpoint receive different tool lists.

Policy enforcement at call time. Scoping the list is not sufficient, because tool arguments carry their own risk. A read-only tool with an unbounded query parameter is not read-only in any meaningful sense. Policy has to evaluate the actual call: which tool, which arguments, which target, under which role.

Data-plane transformation. Results pass through masking and DLP before they are returned toward the client. This is the subject of section 4, and it is the function with the most nuance.

Audit. Every session, every tool call, every result size, every policy decision — recorded in a form that survives an audit and can be shipped to a SIEM.

> A NOTE ON TOOL LIST PERSONALIZATION
>
> MCP supports both the initial `tools/list` request and a `notifications/tools/list_changed` message for updating a client mid-session. In practice, client support for the notification is inconsistent: some clients honor it, some cache the tool list for the lifetime of the connection, some only refresh on reconnect.
>
> The architectural implication is worth stating plainly, because it affects how entitlement changes propagate. **Treat connection-time discovery as the reliable mechanism and mid-session updates as an optimization.** Do not design an authorization model that depends on revocation taking effect within an active session. Revocation must be enforced at call time by the policy engine, not by removing a tool from a list the client may still be holding.

3 The outbound-only connector

The conventional way to give an external service access to an internal system is to expose an endpoint: a reverse proxy, a VPN tunnel, an allowlisted inbound port. Each of those is a change to the network perimeter, each requires a firewall ticket, and each creates a listening surface.

The alternative is to invert the direction. A connector component runs inside the customer network and establishes an outbound, authenticated connection to the control plane. It polls for approved work, executes it locally against the internal system, and returns results over the same encrypted channel.

Three consequences follow, and they are the substance of the design:

1. **No inbound ports, no VPN, no firewall change.** The connector uses the same egress path as any other outbound HTTPS client. This is usually the difference between a two-week network change request and a same-day deployment.
2. **Credentials never leave the network.** The SAP service user, the database password, the API key for an internal service — these are configured in the connector and stored inside the customer's environment. The control plane holds a reference to a credential, never the credential itself.
3. **Payload execution is local.** The query runs where the data lives. What crosses the boundary is the result, after transformation, not the raw dataset.

The trade-off is latency and operational surface: the connector is a component the customer now runs, monitors and updates. That cost is real and should be stated to buyers rather than glossed over.

4 Field-level masking

4.1 Where masking happens

The single most important property is position in the flow. Masking is only meaningful if it is applied **inside the data plane, before the result leaves the customer network** — that is, in the connector, not in the cloud.

Masking applied after data has reached a cloud component protects nothing that mattered. The unmasked data has already crossed the boundary; the transformation is cosmetic. Any architecture claiming masking as a sovereignty control has to be able to point at the exact component where the transformation runs and demonstrate that it sits on the customer's side of the boundary.

4.2 Configuration model

Masking is configured per importer and per field. An importer is a configured data source — an HTTP API, or a database connection (PostgreSQL, MySQL/MariaDB, MS SQL Server, Oracle, MongoDB). For each importer, the fields returned are enumerated and a masking rule is assigned per field.

Rule	Behavior	Use
Redact	Replace with a fixed token	Fields the model never needs (passwords, internal keys)
Pseudonymize	Replace with a stable surrogate, mapping retained locally	Identifiers the model must correlate across records
Partial	Preserve a prefix or suffix	Account numbers, IBANs where the institution matters
Generalize	Reduce precision	Dates of birth to year, postcodes to region
Pass	No transformation	Non-personal business data

The default for a newly configured importer should be **deny-by-default on unclassified fields**, not pass-through. A field nobody has classified is a field nobody has assessed.

4.3 The round trip

Pseudonymization is only useful if the answer can be brought back to the real world. A user asking about overdue invoices needs the actual customer name in the final answer, not `CUSTOMER_7F3A`.

The mapping table therefore lives in the connector, inside the network. Outbound, real values are replaced with surrogates. Inbound, when the model's response references a surrogate, the connector substitutes the real value before it reaches the user's screen. The model reasons over surrogates; the user sees reality; the mapping never crosses the boundary.

This works cleanly for structured fields. It works poorly for free text, which is the next point.

4.4 What masking does not do

This section exists because the credibility of the whole architecture depends on being straight about it.

Masked data is generally still personal data. Under GDPR, pseudonymization is a security measure, not an exemption — pseudonymized data remains within scope of the regulation. Only genuine anonymization, where re-identification is not reasonably possible by any party, falls outside. Because the mapping is deliberately retained in the connector, this architecture produces pseudonymized data, not anonymized data. Any claim that masking alone makes a processing operation compliant is wrong and will be identified as wrong by a competent data protection officer.

Free text defeats field-level rules. A rule on a `customer_name` column does not touch the same name appearing inside a `notes` or `description` field. Pattern-based detection over free text catches common formats — email addresses, IBANs, phone numbers — but it is probabilistic, and recall is never complete. Where free-text fields carry personal data, the honest options are to exclude the field, or to accept a residual risk explicitly.

Combination re-identifies. Individually innocuous fields can identify a person jointly. Masking the name while returning department, role and start date may not protect anyone in a team of six.

Utility and protection trade off directly. Masking a field the model needs to answer the question degrades the answer. This is not a tuning problem to be solved; it is a boundary to be placed deliberately, per use case, with the business owner involved.

Masking is not an access control. It reduces the sensitivity of data that a permitted call returns. It does nothing about a call that should never have been permitted. That is the policy engine's job, and the two must not be confused in a design review.

5 Deployment topologies

Three configurations, differing in where the control plane runs. The connector is present in all three.

Topology A — Managed control plane

The control plane runs as a managed EU-hosted service. The connector runs in the customer network.

- **Crosses the boundary:** transformed results, tool schemas, policy decisions, usage metadata
- **Never crosses:** system credentials, unmasked payloads, pseudonym mappings
- **Customer operates:** the connector only
- **Suits:** most organizations; fastest path to production

Topology B — Self-hosted control plane

The control plane runs in the customer's own cloud tenancy or data center. Nothing routes through the vendor's infrastructure.

- **Crosses the boundary:** nothing operational; software updates and license validation only
- **Customer operates:** control plane and connector
- **Suits:** organizations where third-party processing is itself the blocker, or where a foreign-disclosure argument has to be closed structurally rather than contractually

Topology C — Air-gapped

Topology B with no egress at all. Updates arrive as signed artifacts through the customer's existing offline process. Model access is restricted to models hosted inside the environment.

- **Crosses the boundary:** nothing
- **Suits:** defense, critical infrastructure, and regulated environments with a hard no-egress rule
- **Cost:** manual update process, no telemetry, longer incident diagnosis

COMPARISON

	A · Managed	B · Self-hosted	C · Air-gapped
Credentials in customer network	Yes	Yes	Yes
Unmasked payload leaves network	No	No	No
Configuration held externally	Yes	No	No
Third-party processor involved	Yes	No	No
Customer operational burden	Low	Medium	High
Update path	Automatic	Automatic	Manual, signed
Time to production	Weeks	Weeks to months	Months

The distinction that matters in most procurement conversations is narrower than it first appears. In all three topologies, credentials and unmasked payloads stay inside the network. What differs is whether *configuration and metadata* are held by a third party. That is a genuine question for a regulated buyer, but it is a smaller question than "does our data leave", and framing it accurately shortens the conversation.

6 Request flow

A single request in Topology A, end to end:

1. User authenticates the AI client against the gateway endpoint via the organization's identity provider. The session is bound to a named identity.
2. Client issues `tools/list`. The gateway resolves the identity against the role model and returns only permitted tools.
3. User asks a question in natural language. The model selects a tool and constructs arguments.
4. Client issues `tools/call`. The gateway evaluates the call against policy: is this identity permitted this tool, with these arguments, against this target, right now.
5. On approval, the call is queued for the relevant connector.
6. The connector — already holding an outbound connection — collects the approved call, resolves the credential locally, and executes against the internal system.
7. The result is transformed in the connector: field-level masking, pseudonymization, DLP inspection.
8. The transformed result returns over the existing channel and is passed to the client.
9. The model produces an answer. Surrogates in that answer are resolved back to real values by the connector before display.
10. The gateway records the full transaction: identity, tool, arguments, target, policy decision, result size, timestamp.

Steps 4 and 7 are where the product lives. Step 6 is where the credential boundary holds.

7 Audit, retention, and the works council

Audit logging is straightforward to build and difficult to get right in a German-speaking environment, for a reason that is organizational rather than technical.

A complete audit log of every AI interaction by every named employee is, from a works council's perspective, a system suitable for monitoring employee performance and behavior — which in Germany brings co-determination rights into play under §87 BetrVG. A deployment that ignores this will be stopped after the technical rollout, at the point where it is most expensive to change.

Three design decisions address it:

- **Split the logs by purpose.** Security audit records (identity, tool, target, decision) are retained under a defined policy in the customer's environment. Product usage telemetry sent to the vendor is pseudonymized or aggregated and carries no per-individual behavioral detail.
- **Keep payload-bearing logs local.** Tool-call arguments and results may contain personal data. They belong in the customer environment, subject to the customer's retention policy, not in the vendor's.
- **Define retention explicitly, per log class.** "We log everything indefinitely" is not an answer that survives a data protection review.

Vendors serving this market should be able to hand over a log retention concept and a template works agreement as documents. Claims like *revisionssicher* or *betriebsratsfest* made without those artifacts behind them will not survive first legal review.

8 What a gateway does not solve

Prompt injection. Content retrieved from an internal system can contain instructions the model acts on. A gateway can constrain what tools are callable and log what happened; it cannot make the model ignore adversarial content in a document. Mitigation is architectural — require confirmation for state-changing operations, keep write scopes narrow — not preventive.

Over-permissioned source systems. If the underlying service account is broad, the gateway can narrow it, but it inherits whatever the source system permits. Least privilege has to be established in the source system too.

Model-side retention and training. What the provider does with data it receives is governed by contract and provider configuration, not by the gateway. The gateway reduces what is sent; it does not control what happens after.

The confused deputy. A gateway executing calls on behalf of users is a high-value target: it holds broad connectivity by design. Its own compromise is the dominant risk in the architecture, which is why its authentication, secret handling and update chain warrant more scrutiny than any other component.

Compliance. A gateway supplies technical and organizational measures. Lawful basis, risk classification, records of processing, transparency obligations and AI Act deployer duties remain the operator's

responsibility. Tooling supports the documentation; it does not produce the compliance.

9 Evaluation checklist

Questions worth asking any vendor in this category, including this one:

01	Where exactly does masking execute, and can you show it is on our side of the boundary?
02	What is the default for an unclassified field — pass or deny?
03	How are free-text fields handled, and what is the acknowledged residual risk?
04	Where do system credentials live, and what does the control plane hold instead?
05	Is revocation enforced at call time, or does it depend on the client refreshing its tool list?
06	What precisely is transmitted to you in the managed topology? Ask for a field list, not a category.
07	What is the retention policy per log class, and can we set it?
08	Can you provide a log retention concept and a template works agreement as documents?
09	Which certifications exist today, versus which are in progress? Ask for the certificate.
10	What happens to an in-flight session if the connector loses connectivity?

Appendix — Supported source types

DATABASES

PostgreSQL, MySQL/MariaDB, Microsoft SQL Server, Oracle, MongoDB

BUSINESS SYSTEMS

SAP, Microsoft Dynamics/Navision, Salesforce, ServiceNow

GENERIC

HTTP/REST APIs with configurable authentication

DOWNSTREAM MCP SERVERS

Any compliant MCP server, federated behind the same policy layer

CLIENTS

ChatGPT, Claude, Langdock, Open WebUI, and any MCP-capable client

---

This document describes architecture and is not legal advice. Compliance assessments for a specific deployment require review by qualified counsel and the responsible data protection officer.