ViewLegacy.

Risk Reduction Effects of Not Storing Business Data in the Cloud for Legacy Database Integration

Technical white paper

Published by: ViewLegacy LLC.
Author: ASAHI MASUDA
Publication date: July 6, 2026

This paper evaluates when a legacy database integration architecture that routes business data through a provider cloud, but does not persist that business data there, can reduce security, compliance, operational, and incident-response risk.

1. Executive Summary (1/2)

Legacy database integration often begins with a practical request: give an AI agent, SaaS product, customer portal, partner system, or internal tool access to selected operational data. The risk question is whether that access requires creating another durable cloud-side copy of sensitive business records.

This paper compares six patterns: full replication, ETL or ELT into a warehouse or lake, iPaaS or SaaS synchronization, a custom cloud API with cached records, a non-persistent cloud pass-through gateway, and a capability-based pass-through architecture with an on-premise agent.

The main conclusion is balanced. Not storing business data in the provider cloud can reduce the amount of accumulated sensitive data exposed by cloud-side compromise, backup leakage, log leakage, retention misconfiguration, and downstream analytics sprawl. It can simplify some data retention, deletion, residency, and incident scoping questions.

It does not eliminate risk. Business data still transits the gateway, the on-premise agent still accesses the source database, and operational metadata may still be sensitive. The architecture does not automatically solve legal, contractual, privacy, or security obligations.

1. Executive Summary (2/2)

A non-persistent cloud pass-through architecture is particularly suitable for operational, read-oriented legacy-system access where external systems need specific current records or explicit business capabilities without creating a secondary cloud database.

Examples include:

- retrieving a current customer status for a support agent
- checking invoice status for a workflow
- querying inventory availability for a partner portal
- giving an AI agent a narrow read-only MCP tool instead of raw SQL

It is not always the right architecture. Data replication, data warehouses, governed analytical platforms, or SaaS synchronization may be better when the primary need is analytics, large-scale BI, machine learning, historical reporting, offline availability, or repeated low-latency access to large datasets.

The strongest design is not merely "do not store data." It is: do not store business data in the provider cloud, do not log payloads, do not expose raw SQL, and expose only explicit capabilities with scoped authorization.

2. Background (1/2)

Legacy systems frequently remain the source of truth for customer, order, inventory, accounting, contract, and employee data. They were often built before modern API gateways, SaaS integrations, AI agents, and MCP clients became common access paths.

Modern applications increasingly need access to this data. The common integration answer is to copy it: replicate the database, run ETL into a warehouse, synchronize records into SaaS, cache records in a cloud API backend, or log and trace more data than intended during development and operations.

Once business data is copied into another system, it introduces additional obligations:

- access control, encryption, key management, and operator access review
- backup retention, deletion handling, and data-subject-request workflows
- incident response, audit scope, vendor review, and data residency review
- downstream sharing review for analytics, dashboards, support tools, and exports

Logs, traces, error messages, queues, object storage, temporary files, and dashboards can accidentally become hidden data stores. AI and MCP integrations increase the importance of minimizing what is exposed, stored, and discoverable.

2. Background (2/2)

The evaluation separates data into categories because "no cloud storage" is only meaningful if the boundary is explicit.

Data category	Examples	Treatment in the evaluated pass-through architecture
Business data	customer records, orders, invoices, inventory, contracts, employee data	May transit request/response path; must not be persisted in provider cloud
Operational metadata	timestamp, capability name, caller/API key identity, status, latency, error code, connection status	May be stored for routing, authorization, observability, and audit
Logs	gateway request logs, control events, dashboard views	Must exclude payloads, raw SQL, credentials, sensitive DB errors, and response bodies
Traces	distributed tracing spans, request attributes, sampled payloads	Must not capture request or response payloads
Backups	database snapshots, object storage backups, log archive backups	Must not include business payload stores if the architecture claims non-persistence
Temporary files	retry buffers, crash dumps, exports, debug files	Must not contain business payloads in provider cloud
Dashboards	provider operations UI, support tooling, customer metrics	Should show metadata, not business records

3. Compared Integration Patterns (1/10)

Pattern 1: Full database replication to a cloud database

Business data is copied from the legacy database into a cloud database. The cloud component creates a durable secondary copy, usually including broad tables, indexes, backups, access paths, and administrative tooling.

The main benefit is flexible cloud-side access. AI agents, SaaS products, BI tools, and internal apps can read from a managed dataset without repeatedly touching the source system.

The main risk is accumulated exposure. If the cloud database, its backups, logs, exports, or support tools are compromised, the incident may include a broad secondary dataset. Deletion and retention workflows must cover the source system, the cloud copy, indexes, replicas, backups, and downstream consumers.

This pattern can be suitable when the organization needs durable cloud-side data and has mature controls around replication, retention, encryption, access, monitoring, and incident response.

3. Compared Integration Patterns (2/10)

Pattern 2: ETL or ELT into a data warehouse or data lake

Business data is extracted, transformed, and loaded into an analytical platform. The platform usually stores historical data, derived tables, aggregations, exports, and BI semantic models.

The main benefit is analysis. Warehouses and lakes are strong fits for reporting, historical trend analysis, broad joins, experimentation, and governed BI.

The main risk is analytical sprawl. The data may be copied into staging tables, transformation jobs, object storage, notebooks, feature stores, dashboards, query result caches, and exported spreadsheets. Logs and pipeline error records can also capture row values.

This pattern may be the right answer for analytics. It is usually a poor fit if the only requirement is a narrow operational lookup of one current record.

3. Compared Integration Patterns (3/10)

Pattern 3: iPaaS or SaaS synchronization with stored records

Business data is synchronized into a third-party integration platform or SaaS product. The vendor may store records, connector state, mappings, retry payloads, logs, and support-visible diagnostic data.

The main benefit is delivery speed. Teams can connect systems quickly, often with prebuilt connectors and workflow automation.

The main risk is vendor and downstream scope. Data location, retention, backup, sub-processor, support access, deletion, and logging behavior may depend on the provider's platform. The customer must understand what is stored, where it is stored, who can access it, and how it is removed.

This pattern can be appropriate when the SaaS system legitimately needs its own durable copy. It should not be assumed to be lower risk merely because it is managed by a vendor.

3. Compared Integration Patterns (4/10)

Pattern 4: Custom cloud API backend that caches or stores business records

A custom cloud API exposes a clean interface to external clients, but also stores business records in a database, cache, object store, queue, or search index.

The main benefit is product control. The team can design the API, enforce authentication, tune performance, and hide legacy system complexity.

The main risk is that "cache" often becomes a data store in practice. Cached records may enter backups, debug tools, dashboards, traces, replay queues, and support workflows. Retention and deletion policies must account for both intended and accidental storage.

This pattern can be useful when latency, offline behavior, or heavy repeated access requires cloud-side state. It should not be called non-persistent unless payload storage is technically prevented and tested.

3. Compared Integration Patterns (5/10)

Pattern 5: Non-persistent cloud pass-through API gateway

The cloud gateway authenticates callers, authorizes requests, routes calls, and records operational metadata. Business data passes through the request/response path but is not stored in provider cloud databases, object storage, queues, analytics stores, dashboards, traces, or logs.

The main benefit is reduced accumulated cloud-side exposure. If the provider cloud has no durable business payload store, several breach, backup, retention, deletion, and residency questions become narrower.

The main risk is request-path sensitivity. A compromised gateway can still observe in-flight data. Logs, traces, queues, crash dumps, temporary files, and support tools can invalidate the non-storage claim if they capture payloads.

This pattern fits operational read access where clients need current data but not an analytical copy.

3. Compared Integration Patterns (6/10)

Pattern 6: Capability-based non-persistent pass-through with an on-premise agent

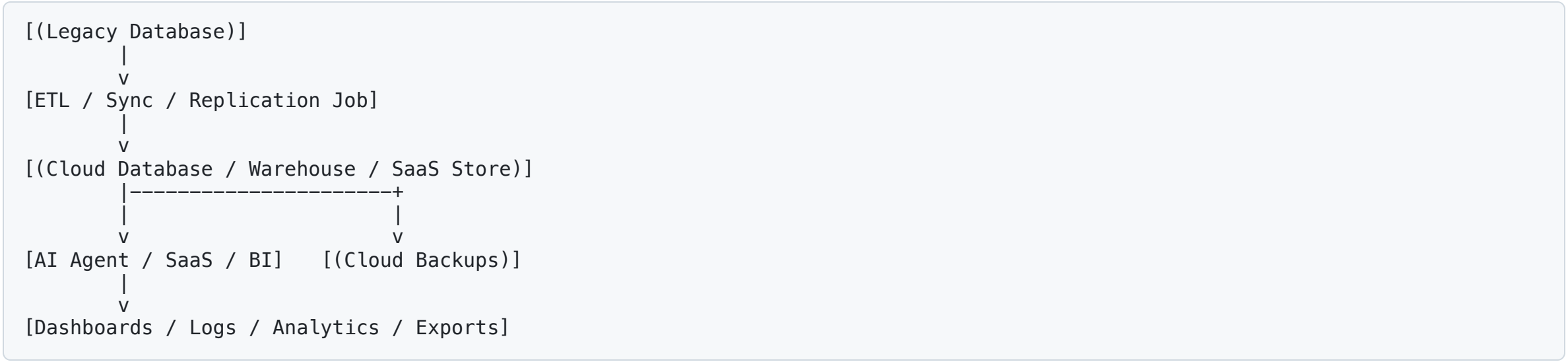
This pattern adds an agent-defined capability boundary. The provider cloud exposes selected read-oriented operations through REST API and MCP, but does not know raw SQL or database credentials. API keys are scoped to allowed capabilities. OpenAPI and MCP tool definitions are filtered by the caller's authorization scope.

The on-premise agent owns SQL, database credentials, validation, execution policy, output filtering, row limits, and business meaning. A `capability.yaml` file defines the operations that can exist.

The main benefit is a narrower discoverable and callable surface. The client sees business capabilities such as `get_invoice_status`, not raw tables, credentials, or broad SQL tools.

The main risk is concentration in the capability and agent layer. If definitions are too broad, output filters are weak, or the agent host is compromised, confidentiality risk remains.

3. Compared Integration Patterns (7/10)



In this model, the integration path intentionally creates a durable secondary dataset. That can be valuable for analytics, performance, and availability, but it expands the cloud-side storage, backup, dashboard, support, and incident-response scope.

3. Compared Integration Patterns (8/10)

Business data still transits the gateway. The risk reduction comes from not accumulating durable business records in provider cloud storage, logs, traces, queues, backups, analytics stores, or dashboards.

3. Compared Integration Patterns (9/10)

```
Provider Cloud
  [Gateway]
    |-- routing
    |-- identity
    |-- authorization
    |-- payload-free observability
    +--> [(Metadata Store)]

Customer Environment
  [On-Prem Agent]
    |-- capability.yaml
    |-- database credentials
    |-- prepared SQL
    |-- validation and output allow-list
    +--> [(Legacy Database)]

Boundary rule:
  Business payloads may transit the gateway.
  Business payloads must not be persisted in provider cloud systems.
```

The boundary must be verified in implementation. It is not enough to say "no storage" if observability, support tools, retries, or backups retain payloads.

3. Compared Integration Patterns (10/10)

Pattern	Durable cloud-side business data?	Typical stored data	Main benefit	Main risk	Best fit
Full database replication	Yes	broad tables, indexes, backups, replicas	flexible cloud access	broad secondary dataset exposure	cloud apps needing durable operational copy
ETL / ELT to DWH or lake	Yes	historical records, derived tables, exports	analytics and BI	analytical sprawl and retention complexity	governed reporting, BI, ML, large joins
iPaaS / SaaS sync	Usually yes	synced records, mappings, connector state, retries	fast integration	vendor storage, support, and downstream scope	SaaS workflows that need local SaaS records
Cloud API with cache	Often yes	cached records, queues, debug data, snapshots	performance and product control	cache becomes another data store	low-latency repeated access
Non-persistent pass-through	Intended no	metadata, auth config, routing state	reduced accumulated cloud payload	transit, logs, traces, queues can still leak	current operational lookup
Capability-based pass-through	Intended no	metadata, capability config, filtered catalogs	narrow discovery and execution	agent and capability controls become critical	read-oriented AI, MCP, SaaS, and internal tools

4. Evaluation Criteria (1/2)

Axis	Full replication	ETL / DWH	iPaaS / SaaS sync	Cloud API cache	Non-persistent pass-through	Capability-based pass-through
Business data persistence in provider cloud	High	High	High	Medium to high	Intended none	Intended none
Size of accumulated sensitive dataset	High	High historical	Medium to high	Depends on cache	Low if controls hold	Low if controls hold
Cloud-side breach blast radius	High	High	Medium to high	Cache-sized	In-flight + metadata	In-flight + metadata + capability scope
Backup and retention burden	High	High	Vendor-dependent	Medium	Lower	Lower
Deletion / DSR complexity	High	High	High	Medium	Lower	Lower
Log and trace leakage risk	High if payloads logged	High in pipelines	High in connectors	High	High if controls fail	High if controls fail
Operational metadata sensitivity	Medium	Medium	Medium	Medium	Medium	Medium
Data residency / vendor review	High	High	High	Medium	Narrower storage review	Narrower storage review

4. Evaluation Criteria (2/2)

Axis	Full replication	ETL / DWH	iPaaS / SaaS sync	Cloud API cache	Non-persistent pass-through	Capability-based pass-through
Audit scope	Broad cloud dataset	Broad analytical platform	Vendor and downstream tools	API plus cache	Gateway metadata and controls	Gateway, metadata, capabilities, agent
Recovery / incident scoping	Complex	Complex	Vendor-dependent	Medium	Narrower if no payload stores	Narrower if capability logs are clean
Operational read access	Good	Sometimes stale	Good if synced	Good	Good	Strong
MCP / AI agent access	Risky if broad	Usually indirect	Depends	Good if scoped	Good if scoped	Strong fit
Analytics and BI	Poor to fair	Strong	Fair	Limited	Poor	Poor unless specifically designed
Operational complexity	Medium	High	Medium	Medium	Medium	Medium to high
Performance / latency	Low after sync	Low for analytics	Low after sync	Low if warm	Higher, source-dependent	Higher, source-dependent

5. Evaluation Method (1/2)

The evaluation is a practical illustrative model, not a formal quantitative risk model, legal opinion, compliance certification, or substitute for a security audit.

The model uses a fictional legacy business system containing customer, order, invoice, inventory, contract, and employee data. It classifies data into five categories:

- public metadata
- internal operational metadata
- confidential business records
- personal data
- sensitive personal or regulated data

For each integration pattern, the model asks what data is stored, transmitted, logged, exposed to dashboards, and included in backups or retry paths. It then evaluates twelve scenarios using qualitative labels such as high risk, medium risk, lower risk, reduced if correctly implemented, depends on controls, or out of scope.

The supporting artifacts are distributed as `src.zip`, which contains `src/evaluation-model.json`, `src/evaluate-scenarios.mjs`, and generated `src/scenario-outcomes.md`.

5. Evaluation Method (2/2)

Data category	Replication / ETL	iPaaS / SaaS Sync	Cloud API with cache	Non-persistent pass-through	Capability-based pass-through
Raw business records	Stored broadly	Stored in vendor system	Stored if cached	Not stored by design	Not stored by design
SQL query results	Stored or query-cached	May be stored as records	Often cached	Transits only	Transits only after output filtering
Request payloads	May enter logs/traces	May enter connector logs	May enter API logs	Must not be stored	Must not be stored
Response bodies	May enter logs/dashboards	May enter SaaS support tools	May be cached/logged	Must not be stored	Must not be stored
Raw SQL	Often in jobs or logs	Usually hidden by connector	May exist in backend	Gateway should not know it	Agent-side only
DB credentials	Connector/cloud-side possible	Connector/vendor-managed	Cloud backend possible	Agent-side preferred	Agent-side only
Operational metadata	Stored	Stored	Stored	Stored	Stored with capability scope
Dashboard data	May show records	May show records	May show cached records	Metadata only	Metadata only
Backups	Include business records	Include synced records	Include cache if stored	Metadata/config only	Metadata/config only

6. Evaluation Results (1/4)

Scenario	Full Replication	ETL / DWH	iPaaS / SaaS Sync	Cloud API Cache	Non-Persistent Pass-Through	Capability-Based Pass-Through
S1 Provider cloud DB compromised	High: secondary dataset exposed	High: historical dataset exposed	High: synced records exposed	Medium/high: cache scope	Reduced if no payload stores	Lower durable-data risk
S2 Object storage backup misconfigured	High	High	Medium/high	Medium	Lower if no payload backups	Lower if metadata/config only
S3 Logs include response bodies	High	High	High	High	High: claim fails	High unless technically blocked
S4 Tracing captures payloads	High	Medium/high	Medium/high	High	High	High unless redacted by design
S5 Queue or retry retains payloads	Medium	High	High	Medium	High: becomes storage	Lower only if retry excludes payloads
S6 Support dashboard exposes records	High	High	Medium/high	High	Lower if metadata-only	Lower if metadata-only and read-only

6. Evaluation Results (2/4)

Scenario	Full Replication	ETL / DWH	iPaaS / SaaS Sync	Cloud API Cache	Non-Persistent Pass-Through	Capability-Based Pass-Through
S7 Deletion or retention enforcement	High complexity	High complexity	High complexity	Medium	Reduced if no payload stores	Reduced; source and metadata remain
S8 Data residency storage review	High scope	High scope	High vendor scope	Medium	Narrower storage review	Narrower; metadata and agent still reviewed
S9 On-prem agent host compromised	Source risk remains	Source connector risk	Connector risk	Connector risk	High: source access	High; reduced only by capability limits
S10 Gateway compromised in transit	High	High	High	Medium/high	Medium: in-flight data	Medium: scoped but not eliminated
S11 Legitimate AI asks for one current record	Works but broad	Works if fresh	Works if synced	Works with cache concerns	Good fit	Strong fit
S12 BI needs historical analytics	Good if governed	Strong fit	Potential fit	Limited	Poor fit	Poor unless designed as analytics

6. Evaluation Results (3/4)

Risk	Storage-based integration	Non-persistent pass-through	Capability-based pass-through
Accumulated sensitive dataset	Usually grows over time through replicas, caches, exports, and backups	Reduced if no provider-cloud payload stores exist	Reduced if capabilities return narrow outputs and payloads are not stored
Cloud-side breach blast radius	Often includes durable business records	Often limited to metadata plus data in flight	Often limited to metadata, tool catalog, capability config, and data in flight
Backup leakage	Backups may contain business records	Lower if backups contain only metadata/config	Lower if capability config excludes payload examples and records
Retention/deletion complexity	Must cover copies, derived tables, indexes, logs, and exports	Source system and metadata policies still required	Source, metadata, capability history, and audit logs still required
Log/trace leakage	Payloads may be logged in many systems	Non-storage claim fails if payloads are logged	Same, plus tool-call arguments must be reviewed
Operator dashboard exposure	Dashboards may show records	Dashboards should show metadata only	Dashboards should show metadata, capability names, status, and latency only
Endpoint/source compromise	Source remains important but cloud copy also exists	Agent and source become critical	Agent, `capability.yaml`, output filtering, and credentials become critical
Analytics suitability	Stronger for analytical workloads	Weak	Weak unless separate governed analytical path exists

6. Evaluation Results (4/4)

The evaluation shows that not storing business data in the provider cloud primarily reduces accumulated-data risk. It reduces durable cloud-side exposure, can narrow breach investigation scope under certain conditions, and can simplify some retention, deletion, backup, and residency questions.

The risk does not disappear. It moves.

In storage-based integration, much of the risk is concentrated in durable cloud datasets, backups, transformations, and downstream consumers. In non-persistent pass-through, the critical controls move to request-path security, agent security, payload-free logging, trace redaction, retry design, dashboard design, support workflows, authorization scopes, and source-system protection.

Operational metadata also remains sensitive. Capability names, caller identity, timing, frequency, status codes, error codes, and latency can reveal business activity patterns even when no business records are stored.

The architecture is therefore a risk reduction strategy, not a guarantee of security, privacy, or compliance.

7. Discussion (1/3)

Not storing business data in the provider cloud reduces one class of risk: durable secondary-copy risk. It can make the question "What business records are stored in the provider cloud?" easier to answer when the implementation truly stores none.

However, the architecture increases the importance of request-path controls. The gateway must authenticate callers, authorize capabilities, reject overbroad access, avoid payload persistence, protect telemetry, and handle errors without leaking data. The agent must validate inputs, enforce policy, use prepared SQL, apply row and byte limits, filter outputs, and protect database credentials.

If logs, traces, queues, error reports, dashboards, temporary files, support tools, or backups capture payloads, the "non-storage" claim becomes false in practice.

Read-only access materially reduces destructive risk, but it does not remove confidentiality risk. A read-only capability can still disclose sensitive records if authorization, row limits, object-level checks, or output filters are wrong.

7. Discussion (2/3)

Non-persistent pass-through is not the best fit when analysts need flexible ad hoc exploration, broad analytical joins across many tables, large historical datasets, ML training data, offline availability, or repeated low-latency access to large datasets.

Data warehouses, data lakes, and governed BI semantic layers may be more appropriate when durable datasets and broad analysis are the actual requirement. In those cases, the right mitigation is not pretending storage is avoidable. It is strong governance: classification, access control, encryption, retention, deletion, lineage, logging, monitoring, and incident response.

Similarly, a SaaS product may legitimately need its own durable record copy to deliver its service. The important question is whether that storage is necessary and governed, not whether all storage is bad.

For operational AI agents and SaaS integrations that need specific current records, capability-based non-persistent pass-through can be easier to govern than full data replication because it exposes explicit operations instead of broad databases.

7. Discussion (3/3)

A "no cloud business data storage" architecture can fail when:

- logs capture request or response payloads
- error reports include record values or raw database errors
- distributed traces capture request bodies, response bodies, headers, or attributes containing sensitive values
- queues, retry buffers, or dead-letter stores persist business data
- dashboards display business records to provider operators
- temporary files, crash dumps, or debug exports are written to cloud disks
- backups include any payload store
- support tools expose payloads to operators
- operational metadata is treated as non-sensitive even though it reveals business activity
- the on-premise agent host is compromised
- the gateway is compromised during request processing
- capability definitions are too broad and return excessive data

The architecture must be tested against these failure modes.

8. Practical Implications (1/2)

Companies connecting AI agents, SaaS tools, or internal systems to legacy databases should first decide whether the use case is operational lookup or analytical replication.

For operational lookup, start with read-only capabilities where possible. Avoid copying full tables unless there is a clear analytical or operational reason. Define capabilities that return only the fields required for the business task.

Keep business data out of logs, traces, dashboards, metrics, queues, error reports, temporary files, and support tools. Store only minimal operational metadata in the provider cloud, and classify that metadata as potentially sensitive.

Use capability-scoped authorization. API keys should be scoped to allowed capabilities. OpenAPI and MCP tool definitions should be filtered based on the caller's authorization scope, so discovery does not reveal unnecessary operations.

Use input validation, row limits, byte limits, timeouts, output allow-lists, encryption in transit, and appropriate secret handling.

8. Practical Implications (2/2)

Security and architecture reviews should explicitly document:

- which business data is stored
- which business data transits
- which business data is never persisted
- which operational metadata is stored
- whether logs include payloads
- whether traces include payloads
- whether backups include payload stores
- whether dashboards show records or only metadata
- whether temporary files, retries, and dead-letter queues can contain business data
- where the on-premise agent runs and how it protects credentials
- how capability definitions constrain inputs, SQL, rows, bytes, and outputs
- how incident response differs for gateway compromise, metadata compromise, and agent/source compromise

Test logging and tracing configurations. Review cloud backups. Inspect support dashboards. Verify that payload-free design is enforced by code and configuration, not only by policy.

9. Limitations and Future Work (1/2)

This evaluation is illustrative and is not a substitute for a full security audit, legal review, privacy review, compliance certification, or customer-specific threat model.

Actual obligations depend on jurisdiction, contracts, data categories, customer policies, provider terms, sub-processors, implementation details, and operational maturity.

The comparison does not benchmark latency, throughput, availability, or cost in depth. It assumes strict controls prevent payloads from being stored in logs, traces, queues, backups, dashboards, monitoring systems, support tools, temporary files, and crash artifacts.

The evaluation focuses primarily on read-oriented operational access. Write operations require additional controls: approval flows, idempotency, transaction design, conflict handling, change logs, rollback procedures, stronger auditing, and clearer responsibility boundaries.

The paper does not fully evaluate human support workflows, privileged operator access, red-team findings, incident-response maturity, data residency law, or third-party subcontractor risk.

9. Limitations and Future Work (2/2)

Future work should include:

- real-world logging tests that prove request and response bodies are not captured
- distributed tracing tests that verify payload redaction
- cloud backup verification to confirm backups do not contain business payload stores
- data-residency analysis for metadata, routing paths, and customer environments
- privacy-impact analysis for personal data and operational metadata
- red-team exercises against gateway, agent, support, and observability workflows
- benchmark comparisons against replication-based approaches
- evaluation of write-capability controls, approval workflows, and audit requirements
- validation of MCP and OpenAPI discovery filtering under different API-key scopes

The architecture should be evaluated as a control set. "No cloud business data storage" is only credible when the implementation also prevents payload logging, payload tracing, raw SQL exposure, excessive capability outputs, and provider support visibility into business records.

10. References (1/2)

- OWASP Cheat Sheet Series, "Logging Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
- OWASP Top 10 Proactive Controls, "C9: Implement Security Logging and Monitoring." https://top10proactive.owasp.org/the-top-10/c9-security-logging-and-monitoring/
- OWASP API Security Top 10 2023, "API1:2023 Broken Object Level Authorization." https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/
- OWASP API Security Top 10 2023, "API3:2023 Broken Object Property Level Authorization." https://owasp.org/API-Security/editions/2023/en/0xa3-broken-object-property-level-authorization/
- NIST, "SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations." https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final
- NIST, "SP 800-92, Guide to Computer Security Log Management." https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf
- Model Context Protocol, "Tools." https://modelcontextprotocol.io/specification/2025-11-25/server/tools
- OpenAPI Initiative, "OpenAPI Specification v3.2.0." https://spec.openapis.org/oas/v3.2.0.html

10. References (2/2)

- OpenAPI Documentation, "Describing API Security." https://learn.openapis.org/specification/security.html
- Amazon Web Services, "Shared Responsibility Model." https://aws.amazon.com/compliance/shared-responsibility-model/
- Microsoft Learn, "Shared responsibility in the cloud." https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility
- Regulation (EU) 2016/679, "Article 5: Principles relating to processing of personal data." https://gdpr-info.eu/art-5-gdpr/
- UK Information Commissioner's Office, "Data minimisation." https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/data-minimisation/
- UK Information Commissioner's Office, "Storage limitation." https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/storage-limitation/

Appendix: Evaluation Artifact Archive

The local illustrative evaluation used for this paper is distributed with the PDF as:

```
src.zip
```

The archive contains the evaluation model, the small generation script, and the generated scenario outcome table:

```
src/evaluation-model.json
src/evaluate-scenarios.mjs
src/scenario-outcomes.md
```

These files record the scenario notes behind the condensed tables in this paper. They are intentionally qualitative and should be read as a repeatable comparison aid, not as a formal probability model.

Finally

This white paper does not argue that non-persistent cloud pass-through is universally superior. It argues that, for read-oriented operational access to legacy systems, avoiding durable cloud-side storage of business data can reduce a specific and important class of risk.

The design is strongest when payload-free logging, payload-free tracing, scoped capabilities, metadata discipline, agent-side enforcement, and source-system security are treated as one control set.

Published by: ViewLegacy LLC.
Author: ASAHI MASUDA
Publication date: July 6, 2026