Data Leakage Prevention Spec

This spec defines the sensitive-data classification scheme, the rules governing when data of each class may move between contexts, the detection and blocking controls for prompt-injection and malicious-retrieval attacks, and the access-logging obligations that feed s3-provenance-metadata-schema. It operationalizes the data-class ceilings declared in t8-permission-architecture into specific runtime controls. Any agent that retrieves external content, processes applicant or guest data, or routes outputs to connectors must implement the guards in this spec.

1. Data classification

Four data classes are defined, matching the authorization ceilings in t8-permission-architecture. Classification is assigned at the source level, not inferred from content at runtime. When the runtime cannot determine the class at ingestion time, the default is restricted.

Class	Definition	Examples
public	Information intentionally published for unrestricted access	Published permit status, STR listing descriptions, public parcel data from the South Walton County GIS layer, publicly posted meeting agendas [VERIFY availability for automated reads under FS Chapter 119]
internal	Information shared within an operating organization with no confidentiality marking	Booking confirmations, internal grant-pipeline notes, operational logs, draft documents not yet sent externally
restricted	Information whose disclosure is bounded by contract, policy, or regulation	Financial records, payout summaries, compliance determinations, audit trails, personal email addresses, user account data
pii	Information that directly identifies or is linkable to a natural person; subject to statutory protections	Guest names + contact data, applicant identity data, payment card data, ePHI; in Florida see FS § 501.171 [VERIFY current version] and FS Chapter 119 exemptions

The classification of a data item is fixed by its source's declared data_class_ceiling in t8-tool-use-control-list. When multiple data classes appear in a single model context window, the entire context is governed by the highest class present. This is the contamination rule: one pii field in the context window means the entire context is treated as pii for egress routing purposes.

2. Cross-context sharing rules

Data moves between five context types: (a) the model context window, (b) scratchpad/memory, (c) tool arguments passed to connectors, (d) outputs delivered to the user interface, (e) provenance store records. Each transition has a rule.

2.1 Model context window

Only public and internal data may enter the model context without redaction. restricted data may enter only after field-level redaction of identifying elements; the redaction event is logged to provenance. pii data must not enter the model context window in raw form under any circumstances. Instead, the agent receives a redacted stub (e.g., GUEST_NAME_REDACTED, APPLICANT_ID_HASH) and the raw data is retained only in the authorized-sink connector.

A model output that includes what appears to be raw PII is flagged as a pii-leak-risk failure mode and routed immediately to a HITL gate before any further action.

2.2 Scratchpad and memory

Scratchpad memory shared across steps within a run is governed by the highest data class in the run. The scratchpad is cleared between runs. If a run involves pii data, the scratchpad is treated as a pii-class store for its entire lifetime and may not be persisted or logged to any sink other than the authorized-sink record in the provenance store.

Cross-run memory (persistent agent memory, vector stores, embedding indexes) must not store restricted or pii class data without an explicit authorized-sink designation and an access-logging hook that writes to s3-provenance-metadata-schema.

2.3 Tool arguments

Tool arguments are subject to the data_class_ceiling declared for the tool in t8-tool-use-control-list. Passing a pii-class argument to a tool whose ceiling is internal is a pii-leak-risk event. The argument is blocked and the event is logged before the call reaches the connector.

2.4 User-facing outputs

Outputs presented to users (HITL review interfaces, dashboards, notifications) must be redacted to the authorization class of the receiving principal. A reviewer in the human-operator role may see restricted data in the HITL review view when necessary to make the review decision; they may not see raw pii unless they hold an explicit pii-authorized-reviewer designation on the canvas.

2.5 Provenance store records

Provenance records in s3-provenance-metadata-schema use evidence_pointer URIs rather than inlining content. This means pii and restricted data is stored once, in the authorized sink, and referenced by a pointer in the provenance record. The pointer itself contains no PII. This is the split-storage rule from C10; violations are gate-failing.

3. Prompt-injection and malicious-retrieval guards

Prompt injection is the primary content-layer threat to agents with tool authority. An attacker who can inject instructions into retrieved content can redirect the agent's next action without compromising the model or the connector.

3.1 Content origin tagging

Every piece of content entering the run must be tagged with its origin class before it reaches the model context:

- agent-instruction: the system prompt and any framework-level instructions. These are the only strings the model may treat as instructions.
- operator-input: human-operator input through the approved HITL interface. Trusted but not instruction-privileged.
- task-data: content retrieved from connectors, RAG stores, or external sources. Never treated as instructions. Only as data to be processed per the agent-instruction.

The origin tag is not a model-level concept; it is enforced by the runtime's context assembly layer before the model sees anything. The model's instructions must explicitly prohibit acting on directives embedded in task-data content.

3.2 Injection detection

An injection detector runs over all task-data content before it is assembled into the context window. The detector checks for:

- Explicit instruction-override patterns: strings that contain keywords like "ignore previous instructions", "you are now", "new system prompt", "disregard", or their obfuscated variants (Unicode lookalikes, encoded sequences, whitespace injection).
- Action-directive patterns: embedded JSON-like structures that match tool-call schemas, suggesting the content is trying to trigger a tool invocation.
- Exfiltration patterns: instructions to summarize and send data to an external endpoint, embed data in a URL, or route output to an unexpected destination.

Detection threshold: any match at or above the classifier's injection-risk threshold triggers the prompt-injection-detected failure mode from s1-failure-mode-register. The affected chunk is quarantined; the raw payload is logged to provenance; the run is halted pending HITL review.

3.3 Malicious retrieval guards

RAG-augmented agents face a related threat: an attacker can insert a poisoned document into the retrieval corpus so that it surfaces in high-relevance results and injects instructions. Guards:

- Source filtering: retrieval results are filtered against the tool allow-list source registry. A chunk whose source does not resolve to a registered source is treated as unknown confidence and blocked from entering the context window.

- Chunk-level injection scan: each retrieved chunk is scanned by the injection detector before assembly. A poisoned chunk is quarantined without blocking the entire retrieval; the run continues with remaining non-poisoned chunks and a lower confidence_band on the affected criterion.
- Citation verification: any citation (statute reference, parcel ID, grant number, URL) included in retrieved content is verified against the source of record before the agent uses it in an output. An unresolvable citation triggers hallucinated-citation per s1-failure-mode-register.

4. Access logging obligations

Every data access event — retrieval, context assembly, tool argument construction, output delivery — emits a log record to s3-provenance-metadata-schema. The log records are the evidence that the data-class rules were enforced, not merely declared.

Event type	Required log fields	Destination
Context assembly	run_id, data_class, source_id, redaction_applied (bool), chunk_hash	provenance store
Tool argument construction	run_id, tool_id, data_class, argument_field_names (not values for restricted/pii), authorized_sink_verified (bool)	provenance store
Injection detection hit	run_id, chunk_id, injection_pattern_matched, disposition (quarantine/allow), confidence_score	provenance store + s1-hitl-review-policy
PII egress block	run_id, data_class, destination, blocking_rule, failure_id=pii-leak-risk	provenance store + s1-hitl-review-policy (raises HITL gate)
User-facing output delivery	run_id, output_class, recipient_role, redaction_applied (bool)	provenance store

Logs are append-only and must not be modified after creation. Log retention follows the canvas-declared retention policy; for Florida public-agency agents, minimum retention is governed by the General Records Schedule [VERIFY GS1-SL or applicable agency schedule under FS Chapter 119].

5. Worked example: STR Guest Data Handling

A guest submits a booking inquiry through the Point Preserve Airbnb listing. The agent receives the inquiry as a webhook payload from OwnerRez. The payload includes: the guest's full name, email address, phone number, the requested dates, and a free-text message from the guest. (illustrative)

Classification step: the payload is classified pii at ingestion (guest name + contact data). The contamination rule applies: the entire payload is pii-class for this run.

Context assembly: the agent is assembling a draft response. It needs the requested dates (public) and a greeting. It must not include the guest's name or contact details in the model context window in raw form. The runtime substitutes {GUEST_FIRST_NAME} (a stub resolved from a redacted lookup) and passes only the dates and the anonymized message text to the model. A context_assembly log record is emitted with redaction_applied: true.

Free-text injection scan: the guest message is scanned by the injection detector before assembly. In this run the message is benign ("Looking forward to the stay — do you allow pets?"). No injection-pattern match; the chunk passes with confidence_band: high.

Draft output construction: the model produces a draft reply. The runtime checks the output for raw PII before it reaches the HITL review queue. No raw PII detected; the draft is staged.

HITL gate: the draft targets gmail-guest-draft, which carries control_level: human-approval in the control list. A HITL gate is opened. The reviewer sees the draft reply (no raw PII), approves it, and the C10 record transitions decision_origin: human-override. The gmail-send tool remains blocked; the reviewer sends from their own Gmail session outside the agent.

Provenance trail: four log records emitted — context assembly (pii, redacted), injection scan (pass), tool argument construction (comms, authorized-sink verified), HITL gate opened and closed. No raw PII appears in any provenance record; all PII is behind the pointer in the OwnerRez authorized-sink record.

6. Maintenance and review

Data-class assignments at the source level are reviewed whenever: (a) a new connector is added to the control list, (b) a regulatory change affects the classification of data the agent handles (e.g., a FS § 501.171 amendment affecting Florida breach-notification scope [VERIFY]), or (c) a pii-leak-risk event fires in production. The injection detection threshold is reviewed as part of the quarterly adversarial-suite expansion in s1-monitoring-rollout-postmortem. New injection patterns discovered in the wild are added to the detection corpus and seeded into the adversarial suite as regression cases.