Unstructured Data Indexing & AI-Query Application — Software Specification (v3)

Document status: Draft for review • **Date:** Aug 7, 2025 • **Owners:** Engineering & Product

---

1. Executive Summary

This system indexes large, unstructured enterprise repositories (Box, SharePoint, OneDrive), applies metadata extraction and optional sensitive-data classification, and exposes natural-language Q&A via a Microsoft Teams bot and an Admin Web UI. It supports incremental scanning, durable progress tracking, **rights-aware (security-trimmed) retrieval**, cost visibility with forecast, and answer rendering including tables and charts.

---

2. Goals & Scope

Goals

- Connect to Box, SharePoint, and OneDrive at enterprise scale via Model Context Protocol (MCP) servers.
- Traverse deep directory trees efficiently; resume and re-scan only when items change.
- Create a master index (MongoDB) and a vector index (Azure AI Search) for retrieval-augmented generation (RAG).
- Provide end-user Q&A in Teams; provide admin controls (policies, cost tracking, directory selection, processing controls) in a Web UI.
- Ensure strong security (least privilege, tenant isolation, secrets in Key Vault) and governance (audit, PII/PHI/ITAR controls).
- **Enforce per-user rights at query time (security trimming)** so users only discover and retrieve data they are authorized to access, across Box and Microsoft 365.

Out of scope

- End-user document editing; DLP beyond the defined policy enforcement; non-Microsoft chat surfaces other than Teams.

---

3. Definitions & Acronyms

- **MCP** — Model Context Protocol: standardized way for tools/servers to expose capabilities to models/clients.

- **MCP Python Interpreter** — MCP tool for controlled execution of Python utilities.
- **Box MCP Server** — MCP server providing Box file system tools.
- **Microsoft files-mcp-server** — MCP server exposing SharePoint/OneDrive file system tools.
- **RAG** — Retrieval Augmented Generation.
- **PII/PHI/PCI/ITAR** — Sensitive data categories.
- **Security trimming** — Restricting query scope and results based on the caller's effective permissions.
- **OBO (On-Behalf-Of)** — OAuth flow where a service exchanges a user token for downstream resource access.

---

4. System Architecture

4.1 Components

- **Admin Web UI** (Next.js or equivalent): Admin policy editor, directory picker, run controls, spend dashboard, answer audit, **connection manager** for user-level provider consents.
- **Teams Bot Service** (Python/Node): Conversational entry point; Adaptive Cards; OAuth via Entra ID; **prompts users to connect providers (Box, M365) and caches per-user tokens**.
- **RAG Orchestrator** (Python): Query parsing, **principal resolution**, retrieval with security filter, prompt assembly, post-processing (tables/graphs) and citations.
- **Ingestion & Monitor Service** (Python): Schedules full/incremental scans; receives change events; orchestrates per-file pipelines; **captures ACLs from providers**.
- **Authorization (AuthZ) Service** (Python): **Resolves effective principals** for the caller (user object ID/UPN, Entra groups, Box user/groups/roles), computes a filter for the index, and performs **final allow/deny** checks.
- **MCP Connectors** (sidecars): MCP Python Interpreter, Box MCP Server, Microsoft files-mcp-server.
- **Pre-filter & Metadata Extractor** (Python): File type detection, text extraction, metadata normalization.
- **Summarizer & Embedder** (Azure OpenAI): Per-file summary JSON and embeddings.

- **Vector Index** (Azure AI Search, vector + keyword): Stores embeddings and metadata for retrieval, **including security principals**.
- **Master Database** (MongoDB): File catalog, processing state, **ACL snapshots**, policies, costs, user connections, Q&A logs, **security decisions**.
- **Object Store** (Azure Blob): Cache for large artifacts, chart images, CSV exports, backups (immutable/WORM buckets).
- **Secrets & Config** (Azure Key Vault); **Observability** (Azure Monitor/App Insights); **CI/CD** (GitHub Actions/Azure DevOps).

4.2 Data Flow (high level)

1. Admin connects sources (Box/SharePoint/OneDrive). 2) Ingestion traverse discovers files and emits work items. 3) Pre-filter extracts text/metadata and **captures provider ACLs**. 4) Summarizer creates structured summary JSON; Embedder produces embeddings. 5) Index update in MongoDB + AI Search **including allowed principals**. 6) User queries in Teams → Bot obtains **user identity and per-provider tokens (OBO)** → AuthZ resolves principals → Orchestrator queries the index with a **security filter** → Post-process with tables/graphs, **trim/censor** where required. 7) Costs metered and forecasted; policies enforced.

4.3 Tenancy & Isolation

Single-control plane, per-tenant credentials and policy records. Data partitioning at DB/Index level via tenant_id and at storage via per-tenant containers. **Security trimming is always evaluated per-tenant and per-user.**

---

5. Detailed Component Specifications

5.1 Ingestion & Monitor Service

Responsibilities: discovery, change detection, enqueueing, orchestration of file pipeline, poisoning/redo handling, metrics, **ACL capture**.

5.1.1 Directory Traversal Controller (Python)

Directory traversal is **triggered and controlled by an associated set of Python programs** that operate in three modes:

- **CLI** — python -m ingestion.traverse --tenant TENANT --source {box|sharepoint|onedrive} --mode {full|incremental} [--paths "/Finance,/HR"]

- **Scheduler** — CronJob (AKS) for periodic incremental scans (default hourly, tenant-configurable).
- **Event-driven** — Webhook/MCP event ingester maps provider change notifications to targeted re-scan queue items.

The controller uses MCP tools for listing and content fetch (Box MCP Server, files-mcp-server). It persists traversal cursors (folder IDs, delta tokens), last-seen ETags/versions, and content hashes in MongoDB.

5.1.2 Change Detection, ACLs & Idempotency

- A file is re-processed only if any of: **modified_time**, **provider version/ETag**, **content hash**, or **ACL hash** has changed.
- During ingestion, the service captures **provider ACLs** for each item:
    - **Box**: collaborators (users/groups), roles (viewer, editor, etc.), shared link scope.
    - **SharePoint/OneDrive**: inheritance, unique permissions, sharing links, role assignments.
- The ACL snapshot is normalized and hashed; stored with each file record and propagated to the index as **allowed_principals[]** and **allowed_groups[]** (provider-scoped IDs).
- A per-file **processing_state** document records last successful stage, retries, error snapshots, and the last ACL hash processed.

5.1.3 Throughput & Backpressure

Work queues sized per tenant; HPA scales workers by queue depth/latency. Max parallelism per provider to respect API limits.

5.2 Pre-filter & Metadata Extraction

Normalizes provider metadata (file_id, path, size, mime, author, modified_time, permissions) and extracts text (Office/PDF). **Progress recording** updates **processing_state** with version/hash/ACL hash. Files are not re-scanned unless these values change. Emits a canonical document for downstream summarization and embedding.

5.3 Sensitive Data Classification & Policies

5.3.1 Admin-Configurable “Sensitive Data” Definition (UI-Driven)

Admin UI provides a **policy editor** to define what “sensitive data” means per tenant:

- **Categories**: PII, PCI, PHI, ITAR/EAR, Trade Secrets, Legal Privilege, HR.

- **Detectors**: Enable/disable built-in regex/patterns or NLP classifiers; define **custom regexes**.
- **Examples/Exclusions**: Provide examples (e.g., SSN formats) and **explicit exclusions** (e.g., employee IDs like E-12345).
- **Enforcement**: Choose behaviors on retrieval/answer: mask, block, alert, or allow with banner. Policies are versioned and stored in MongoDB; a policy hash is stamped into each file's metadata during classification.

5.3.2 Optional Directory Selection

This feature is **optional**. When enabled, Admins see a **browsable directory tree** (via the MCP providers) and can select one or more folders to prioritize classification/summarization, restrict Q&A scope, or trigger ad-hoc re-processing for the selected subtrees.

5.4 Summarization & Embeddings

- **Summaries**: Azure OpenAI (GPT-4-class) produces a compact JSON summary with fields: title, purpose, entities, dates, table of contents (if any), and key facts.
- **Embeddings**: Azure OpenAI text-embedding model. Chunking by semantic boundaries (target 1–2k tokens).
- **Index Update**: Upsert to Azure AI Search (vector + keyword) with file_id, tenant_id, path, sensitivity flags, **allowed_principals/groups**, and summary facets.

5.5 Authorization (Security Trimming) & RAG Orchestration

5.5.1 Identity & Token Flow

- **Teams auth**: Bot authenticates the user with Entra ID.
- **On-Behalf-Of**: Orchestrator exchanges the Teams token for Microsoft Graph (to read group membership) and for SharePoint/OneDrive access tokens, as configured.
- **Box consent**: Users are prompted (Admin UI or Bot card) to sign into Box (per-user OAuth). Tokens are stored encrypted (Key Vault + DPAPI/Azure Managed Identity) and rotated.
- **Identity mapping**: The system maintains a mapping from **Teams UPN → Entra object ID / Group IDs → provider identities** (Box user ID, M365 UserPrincipalName, etc.).

5.5.2 Security Filter Construction

- At query time, the **AuthZ Service** resolves the caller's **effective principals**:
    - Direct user ID for each provider (e.g., box:user:12345).

    - Group memberships (e.g., sp:group:{GUID}, box:group:engineering).
    - Special scopes from sharing links (organization-wide, anonymous links) if allowed by policy.
- The Orchestrator queries Azure AI Search with a **filter expression**: *return only documents where any of the caller's principals intersect allowed_principals / allowed_groups*.
- **Final gate**: Before returning snippets, the Orchestrator performs a **per-item allow/deny** check via AuthZ using the latest provider tokens (to handle race conditions and recently changed ACLs).

5.5.3 Retrieval, Prompting & Rendering

- Retrieval uses **security-filtered** vector + keyword search.
- Prompt construction includes only content the user can access; counts/statistics and "not found" messages are **security trimmed** to avoid information disclosure.
- **Answer Rendering** (tables/graphs) only includes rows derived from permitted items. If the question targets restricted areas, the bot returns: *"Some requested data is restricted for your account."*

5.5.4 Caching & Performance

- Per-user **principal cache** with short TTL (e.g., 5–15 minutes) keyed by tenant + user object ID and provider.
- Index filters push down most trimming to Azure AI Search; **final gate** is limited to top-k results.

5.6 Cost Metering & Forecast Service

(unchanged) Meters tokens/ops/runtime; aggregates and forecasts; exposed to Admin UI.

5.7 Admin Web UI

- **Spend Dashboard**: To-date, MTD, forecast; per-feature; budgets & throttling.
- **Policy Editor**: Manage sensitive data definitions, examples, and exclusions (5.3.1).
- **Directory Picker**: Optional tree view to select/prioritize folders (5.3.2).
- **Run Control**: Start/stop ingestion, force incremental/full scans, re-process failures.
- **Connections**: User page to connect Box and renew M365 consent; displays connected status and last refresh.

- **Answer Audit**: Searchable Q&A history; view tables/graphs and download CSV payloads; **security decision trail** (why a document was included/excluded).

5.8 Data Model (MongoDB)

Collections & key fields (**new fields in bold**):

- files: tenant_id, file_id (provider + id), path, mime, size, author, modified_time, version/etag, content_hash, **acl_hash, allowed_principals[], allowed_groups[]**, sensitivity_flags, summary_ref.
- processing_state: file_id, stage (prefilter/summarize/embed/index), last_success_ts, retry_count, error_snapshot, cursors, **last_acl_hash**.
- policies: tenant_id, version, definitions {categories, regexes, exclusions, behaviors}, created_by, created_at.
- user_connections: tenant_id, user_object_id, upn, **provider_identities** (box_user_id, m365_upn), **token_refs** (Key Vault secrets), updated_at.
- identity_cache: tenant_id, user_object_id, **principal_set**, **expires_at**.
- cost_usage: ts, tenant_id, feature (ingest/summarize/embed/query), units, unit_cost, amount.
- qa_logs: question, answer, citations[], table_json_ref, chart_ref, csv_ref, cost_snapshot, **security_trace** (principals used, filter applied, final gate decisions).
- embeddings: file_id, chunk_id, vector_ref/index_id, metadata. Indexes: compound on (tenant_id, file_id); (tenant_id, path); (tenant_id, **allowed_principals**); TTL on identity_cache; TTL on transient error logs.

---

6. External Systems & Connectors

- **MCP Python Interpreter**: Executes vetted traversal/util scripts in a sandbox. Tool whitelist only.
- **Box MCP Server**: Folder listing, file metadata/content fetch, delta streams, **ACL capture**.
- **Microsoft files-mcp-server**: SharePoint/OneDrive folder tree, file metadata/content, delta queries, **ACL capture**.
- **Microsoft Graph**: Group membership resolution; OBO token exchange; SharePoint/OneDrive access.
- **Box OAuth**: Per-user tokens for Box access.

- **Azure OpenAI**: Chat/completions for summarization and QA; embeddings for retrieval.
- **Azure AI Search**: Vector + keyword hybrid retrieval; **metadata filters for security trimming**.
- **Azure Key Vault**: Secrets, connection strings, certificates, user token references.
- **Azure Blob Storage**: Artifacts, chart images, CSVs, and immutable backups.

---

7. Security, Privacy & Compliance

- **AuthN/Z**: Entra ID; Admin UI RBAC (Owner, Operator, Auditor). Bot uses user identity; per-tenant authorization enforced in orchestrator filters; **final allow/deny** before output.
- **Data Protection**: AES-256 at rest; TLS 1.2+ in transit; KMS-managed keys.
- **Sensitive Data Enforcement**: Policies applied at retrieval time and answer rendering; masking and blocking rules logged.
- **Security Trimming Guarantees**: No disclosure via counts, snippets, or citations for resources the caller cannot access. Side channels (error messages, timing) considered.
- **Auditability**: Immutable Q&A logs with input/output, citations, policy decisions, **security traces**.
- **Data Residency**: Region selection per tenant; co-locate AI Search/OpenAI.

---

8. DevOps, IaC, and Repository Layout

8.1 Repo Structure

```
/docker/
  bot/Dockerfile
  orchestrator/Dockerfile
  ingestion/Dockerfile
  prefilter/Dockerfile
  ai-pipeline/Dockerfile
  admin-ui/Dockerfile
  mcp-box-server/Dockerfile

  mcp-files-server/Dockerfile

  authz/Dockerfile


/infra/

  azureai/provision_azure_openai.bicep

  azureai/provision_ai_search.bicep

  azureai/provision_key_vault.bicep

  k8s/*.yaml  # Deployments, Services, HPA, CronJobs


/scripts/

  teams/register_bot.py

  teams/configure_channels.py

  azure/set_env_keyvault.sh

  azure/provision_ai.sh

  cost/export_usage_report.py

  cost/recalculate_forecast.py

  ingestion/run_traverse.py

  ingestion/enqueue_webhook_event.py

  authz/seed_user_mapping.py

  backup/run_backup.sh

  restore/run_restore.sh
```

8.2 Pipelines

- Build & push Docker images; run unit/integration tests; deploy to AKS (dev→stg→prod).
- Secrets from Key Vault via federated identity; no secrets in pipelines.

- **Security tests**: Negative tests ensure no unauthorized items leak through answers or counts.

8.3 Environments

- **Local**: docker-compose.yaml spins up all services + local Mongo + Azurite (for dev).
- **AKS**: Namespaces per stage; HPA by CPU and queue depth; CronJobs for backups and index reconciliation.

---

9. Deployment & Configuration

- **Teams Bot**: Scripted registration (/scripts/teams/register_bot.py); outputs App ID, password, and messaging endpoint; manifest packaged and published; optional /scripts/teams/configure_channels.py to install to Teams/Channels.
- **Azure AI Services**: provision_azure_openai.bicep & provision_ai_search.bicep; /scripts/azure/provision_ai.sh wires resources, alerts, and stores endpoints/keys in Key Vault.
- **MCP Servers**: Deployed as sidecars with health checks; provider credentials injected at start.
- **AuthZ Service**: Deployed with access to Key Vault and provider SDKs; caches principal sets; exposes /resolve and /whoami endpoints.
- **Configuration**: Per-tenant JSON (policies, directory preferences, budgets) stored in DB; environment configuration via Key Vault references.

---

10. Observability & SRE

- **Metrics**: Queue depth, throughput, failure rates, scan latency, token usage, cost per feature, **security-trim hit/miss rate**, principal cache hit rate.
- **Logs**: Structured JSON (OpenTelemetry); correlation IDs across components.
- **Tracing**: Ingestion pipeline spans; RAG request spans including retrieval and LLM calls; **security filter construction and final gate spans**.
- **Dashboards & Alerts**: Spend thresholds, ingestion stalls, elevated 5xx, policy violations, **unauthorized-access attempts**.

---

11. Performance & Scalability Targets

- Ingestion: ≥ 1M files/day per region with p95 stage latency < 2s/file for text-extractable formats.
- Q&A: p95 time-to-first-token < 3s; p95 total answer < 10s for top-k=8 with security filters.
- **AuthZ**: Principal resolution p95 < 200ms (cached), < 1s (cold).
- Cost: Maintain cost per ingested file below configured threshold; automatic throttling when budget near limit.

---

12. Testing & Quality

- **Unit**: Policy engine, changelog logic, summarization prompts (golden tests), cost aggregation, **security filter builder**.
- **Integration**: MCP connector flows; ACL capture; AI Search upserts with allowed principals; AuthZ /resolve end-to-end; Teams bot roundtrips.
- **E2E**: Seeded repositories; verify incremental scan correctness, policy enforcement, **security trimming**, and Q&A outputs (tables/graphs included).
- **Security**: Secret-scan, SAST, container image scan, dependency audit, **negative tests for access leakage**.

---

13. Risks & Mitigations

- **Provider API limits** → Adaptive rate limiting; retry with jitter; backoff and reschedule.
- **LLM hallucinations** → Strict citation requirement; retrieval-only facts; confidence banners; guardrails.
- **Cost overrun** → Budgets, alerts, throttle batch jobs; forecast visible in UI.
- **Data leakage** → **Defense-in-depth**: index-time filters + final gate + redaction + audit.
- **Stale ACLs** → Short-TTL principal caches; include ACL hash in change detection; final gate against providers.

---

14. Roadmap (post-v3)

- Additional connectors (Google Drive, S3 via MCP equivalents).
- Fine-tuned classifiers for domain-specific sensitivity.
- Report builder for recurring analytics across repositories.
- **Attribute-based access control (ABAC) support** and policy simulation ("what-if") tooling.

Appendix A — Teams App Manifest (template)

- Name, ID, bot endpoints, permissions, valid domains, OAuth settings.

Appendix B — API Endpoints (selected)

- POST /ingestion/start|stop — admin-auth only.
- POST /ingestion/rescan — {paths[], mode}.
- GET /cost/summary — to-date, MTD, forecast.
- POST /policy — create/update policy.
- POST /bot/notify — admin broadcast.
- **AuthZ**: GET /authz/resolve?upn=..., GET /authz/whoami.
- **Query**: GET /query (legacy), GET /query_secure?upn=... (security-trimmed; Teams uses caller token instead of query string in production).

Appendix C — Answer Table/Graph JSON Schema

- columns[], rows[]; chart { type, x, y, series }.

Appendix D — Cost Calculation

- Token usage × model unit costs; AI Search Ops × tier rate; container-hours × node price; OCR per page.

Appendix E — CLI Reference

- run_traverse.py (options), enqueue_webhook_event.py (payload), export_usage_report.py (filters).
- authz/seed_user_mapping.py (local dev seeding of user→principals for tests).