Python Ingestion Planner Specification

1. Purpose & Scope

Purpose: Generate and enqueue ingestion work items (messages) for multiple providers on a schedule.
Out of scope: Actual data download/ETL (handled by worker containers) and Databricks transformations.

2. High-Level Design

Runs as an Azure Container Apps (ACA) Job (cron). Reads provider configurations and state from Azure SQL Database,
computes ingestion windows, sends one Service Bus message per unit of work, and updates metadata in SQL for traceability.

3. Azure Resources

- Service Bus (namespace + queue: download-requests)
- Azure SQL Database (provider metadata + state)
- Optional Key Vault (secrets)
- Azure Container Apps environment
- Application Insights for logs and metrics

4. Runtime & Packaging

- Python 3.11+, containerized (Docker)
- Entrypoint: python -m planner.main
- Dependencies: azure-identity, azure-servicebus, pyodbc, tenacity, pydantic, structlog, python-dateutil
- Configuration via environment variables

5. Configuration Variables

SB_NAMESPACE, SB_QUEUE, SB_AUTH, SQL_SERVER, SQL_DB, SQL_AUTH, SQL_USER, SQL_PASSWORD,
PLANNER_TIMEZONE, DEFAULT_WINDOW_MINUTES, MAX_MESSAGES_PER_RUN, LOG_LEVEL

6. Database Schema (Azure SQL)

Schema: ingest
Tables:
- providers: catalog of data sources and cadence
- provider_state: last successful/planned run timestamps
- runs: audit trail of planned/enqueued windows

7. Message Contract (Service Bus)

Queue: download-requests
Example JSON body:
{
"provider": "mlsgrid",
"task_id": "mlsgrid-2025-10-24T10:00-11:00Z",
"range": {"from": "2025-10-24T10:00:00Z", "to": "2025-10-24T11:00:00Z"},
"attempt": 0
}

8. Planner Logic Flow

1. Load providers from SQL (enabled only)
2. Determine time window using provider cadence and last watermark
3. Slice by time/page if needed
4. Enqueue one message per unit of work
5. Record runs and update provider_state
6. Skip if already up to date

9. Error Handling & Retries

- Use tenacity for transient SQL/Service Bus errors (exponential backoff)
- Log failures and mark failed runs
- Never advance watermark on failure
- Logs include provider, window, attempt, and error message

10. Observability & Security

- Structured JSON logs (App Insights)
- Metrics: messages enqueued, runs skipped, runs failed
- Managed Identity preferred for Service Bus and SQL
- Secrets stored in Key Vault

11. Deployment Parameters

- Platform: Azure Container Apps Job
- Schedule: */15 * * * * (every 15 minutes)
- Resources: 0.25 vCPU, 1GB RAM
- Retries: 3 attempts, timeout 10 minutes

12. Acceptance Criteria

- Provider state updated correctly
- Correct number of Service Bus messages per run
- SQL run audit updated with proper timestamps
- Graceful failure logging with retries
- Logs and metrics visible in App Insights

13. Developer Deliverables

Project structure:
planner/
main.py
config.py
sql_repo.py
sb_client.py
slicer.py
models.py
Dockerfile
requirements.txt
tests/
README.md