Microservice Patching Onboarding Document

Purpose

This document is a handoff guide for onboarding new agents into the microservice patching workflow. It includes:

- A log of services already patched.
- Guidelines for strict patch formatting.
- Boilerplate requirements for consistency.

✅ Services Already Patched

1. **ScannerMS**
   - Updated decorator with `dependencies` and `side_effects`.
   - Class now inherits `BaseService`.
   - Added `super().__init__("ScannerMS")`.
2. **ContextAggregatorMS**
   - Fixed decorator service name.
   - Added `dependencies` and `side_effects`.
   - Class now inherits `BaseService`.
   - Corrected `__init__` indentation and added `super().__init__`.
3. **DiffEngineMS**
   - Fixed decorator service name.
   - Added `dependencies` (`sqlite3`, `difflib`, `uuid`, `datetime`).
   - Declared `side_effects` (`db:read`, `db:write`).
   - Class now inherits `BaseService`.
   - Added `super().__init__`.
4. **ExplorerWidgetMS**
   - Fixed decorator service name.
   - Added `dependencies` (`tkinter`, `ttk`).

    - Declared `side_effects` (`ui:update`, `ui:read`, `filesystem:read`).
    - Class now inherits `BaseService`.
    - Added `super().__init__`.
5. **FingerprintScannerMS**
    - Fixed decorator service name.
    - Added `dependencies` (`hashlib`, `os`).
    - Declared `side_effects` (`filesystem:read`).
    - Class now inherits `BaseService`.
    - Added `super().__init__`.
6. **GitPilotMS**
    - Fixed decorator service name.
    - Added `dependencies` (`git`, `subprocess`, `tkinter`).
    - Declared `side_effects` (`filesystem:read`, `filesystem:write`, `network:outbound`, `ui:update`).
    - Fixed broken class declaration.
    - Class now inherits `BaseService` and `ttk.Frame`.
    - Added `super().__init__`.
7. **NeuralGraphEngineMS**
    - Fixed decorator service name.
    - Added `dependencies` (`pygame`, `math`, `random`).
    - Declared `side_effects` (`ui:update`, `render:write`).
    - Class now inherits `BaseService`.
    - Added `super().__init__`.
    - Added `side_effects` to endpoints (`set_data`, `step_physics`, `get_image_bytes`).
8. **NeuralGraphViewerMS**
    - Decorator updated with `dependencies` (`tkinter`, `PIL`, `sqlite3`, `json`).
    - Declared `side_effects` (`ui:update`, `filesystem:read`).
    - Class now inherits `BaseService` and `ttk.Frame`.
    - Added `BaseService.__init__`.
    - Added `side_effects` to endpoints (`run_search`, `animate`).
9. **LogViewMS**
    - Fixed decorator service name.
    - Added `dependencies` (`tkinter`, `logging`, `queue`).
    - Declared `side_effects` (`ui:update`, `filesystem:write`).

    - Class now inherits `BaseService` and `tk.Frame`.
    - Added `BaseService.__init__`.

⚙️ Patch Guidelines

1. **Decorator Requirements**
    - Always update `@service_metadata` with:
        - Correct service name (match class name).
        - `dependencies` (all imported libraries).
        - `side_effects` (filesystem, db, ui, network, render, etc.).
    - Ensure indentation is consistent (4 spaces).
    - End lines with Windows newlines (`\r\n`).
2. **Class Requirements**
    - All services must inherit from `BaseService`.
    - If UI-based, inherit from both `BaseService` and the UI class (e.g., `ttk.Frame`).
    - Always call `super().__init__("ServiceName")` in `__init__`.
3. **Endpoint Requirements**
    - Every `@service_endpoint` must declare `side_effects` explicitly.
    - Inputs/outputs must be JSON-compatible types.
    - Indentation must be 4 spaces.
4. **Patch Format**
    - Use atomic JSON hunks.
    - Each hunk must:
        - Include `description`.
        - Provide exact `search_block` and `replace_block`.
        - Preserve indentation and newlines.
    - Never bundle multiple unrelated changes in one hunk.

🚦 Next Agent Instructions

- Continue patching remaining services following the above strict guidelines.
- Verify each decorator block matches exactly before replacing.

- Be strict about boilerplate: `BaseService` inheritance, `super().__init__`, explicit `dependencies` and `side_effects`.
- Keep a running log of patched services in this document.
- When the 20-file limit is reached, start a new conversation and continue onboarding.

📌 Reminder

This document is the authoritative onboarding guide. Treat it as the contract for patching consistency across all microservices.

Would you like me to also add a **“Next Services To Patch”** section so the next agent knows exactly which files to tackle first (e.g., `MonacoHostMS`, `ChunkingRouterMS`, etc.)?