File I/O & Local LLM Workspace Integration

When running a Local LLM within the Hearthlink platform, you can grant it controlled access to a predefined workspace on disk. Here's how:

---

1. Designate a Workspace Directory

- Choose a directory under the user data path (e.g. `%APPDATA%\\Hearthlink\\workspace` on Windows, `~/.hearthlink/workspace` on Unix).
- Ensure the directory is created on startup by the Launcher Stub or Context Manager.

2. Expose File I/O via an API Layer

- **Context Manager Service** should implement REST endpoints for file operations within this workspace:
- `GET /workspace/files` → list files
- `GET /workspace/files/{path}` → read file content
- `POST /workspace/files/{path}` → write or update file
- `DELETE /workspace/files/{path}` → remove file
- `POST /workspace/files/{path}/rename` → rename or move
- Enforce path sanitization: only allow operations within the `workspace` root to prevent directory traversal.
- Authenticate and authorize each call via OAuth2 scopes (e.g. `workspace.read`, `workspace.write`).

3. Integrate with Local LLM Backend

- When spawning the LLM process (e.g. Ollama, Llama.cpp), pass the workspace path as an environment variable (e.g. `WORKSPACE_DIR`).
- Use a privileged **Preload Script** (Electron) or **IPC** (WinUI) to expose file APIs to the renderer:

```
// In preload.js
const { ipcMain } = require('electron');
ipcMain.handle('workspace-read', (_, relPath) => api.getFile(relPath));
ipcMain.handle('workspace-write', (_, relPath, content) =>
api.writeFile(relPath, content));
```

- In your LLM integration library, call these file APIs to load datasets, save outputs, and manage session artifacts.

4. Security & Audit

- Every file operation emits an audit event to Vault:

```
{ "event":"file.write", "path":"notes.txt", "actor":"alice-session-123",
"timestamp":"..." }
```

- Store audit logs encrypted in Vault's AuditLog table.
- Enforce RBAC: only users with `workspace.write` can write or delete; `workspace.read` suffices for read/list operations.

5. Example Usage Flow

1. **User** invokes `!save analysis.txt` in chat.
2. **Alice** calls `POST /workspace/files/analysis.txt` with content.
3. **Context Manager** writes file under the workspace directory.
4. **Context Manager** emits audit entry to Vault.
5. **LLM** can later load `analysis.txt` via `GET /workspace/files/analysis.txt` during generation.

This pattern ensures your Local LLM has flexible file I/O within a sandboxed environment, with full auditability and access control.

End of De minimus spec.