Makefile constants reference

Here's a **focused checklist** for variable substitution opportunities in your Makefile. I'll highlight the **biggest wins first** (high-impact changes) and then secondary optimizations.

High-Impact Opportunities

1. NPM / NPX Commands

- **Current pattern**: You run `npx openapi-to-postmanv2`, `npx @stoplight/prism-cli`, etc., directly in different targets.
- **Why change**: You've already defined variables like `GENERATOR_OFFICIAL`, `PRISM`, `NEWMAN`, `REDOCLY`, etc.
- **Action**:
  - Ensure **all `npx ...` calls** are replaced with these variables.
  - Consider a `NPM_GLOBAL_TOOLS` or `NPM_TOOLS` variable (like we discussed for `install`).

---

2. JQ Scripts

- **Current pattern**: `$(SCRIPTS_DIR)/fix_paths.jq`, `merge.jq`, etc., are referenced directly.
- **Why change**: If you ever reorganize `$(SCRIPTS_DIR)`, you'll need to change them everywhere.
- **Action**: Create variables like:

```makefile
FIX_PATHS := $(SCRIPTS_DIR)/fix_paths.jq
MERGE_JQ  := $(SCRIPTS_DIR)/merge.jq
ADD_TESTS := $(SCRIPTS_DIR)/add_tests.jq
```

  Then replace calls in targets with `$(FIX_PATHS)` etc.

---

3. Postman Paths

- **Current pattern**: Many references to `postman/generated/` and `postman/custom/`.
- **Why change**: These appear multiple times (e.g., `COLLECTION_RAW`, `COLLECTION_FIXED`, etc.).
- **Action**: You already have `POSTMAN_GEN_DIR` and `POSTMAN_CUSTOM_DIR` — make sure **all Postman file paths** use those variables (you already have good coverage here but worth checking).

---

4. Base URLs and Token Extraction

- **Current pattern**:

```makefile
BASE_URL_RAW := $(shell [ -f $(ENV_FILE) ] && jq -r '.environment.values[] |
select(.key=="baseUrl") | .value' $(ENV_FILE))
```

- **Why change**: These `jq` calls might be repeated in other parts of your Makefile or scripts.
- **Action**: If used in multiple places, consider a dedicated variable or **function-style variable**.

Secondary Opportunities

5. Python Environment

- **Current pattern**:
  `$(PYTHON_ENV_DIR)/e2o.venv/bin/python` and `pip` are referenced via `VENV_PYTHON` and `VENV_PIP`.
- **Action**: Confirm all Python-related commands use these variables. If you install packages, define:

  ```makefile
  PIP_INSTALL := $(VENV_PIP) install
  ```

  and call `$(PIP_INSTALL) -r requirements.txt`.

---

6. Brew Install Tools

- If you plan to add more Homebrew packages, you can define:

  ```makefile
  BREW_TOOLS := openapi-diff
  ```

  Then:

  ```makefile
  brew install $(BREW_TOOLS) || echo "..."
  ```

---

7. Newman HTML Reports

- **Current pattern**: `newman run ... --reporter-html-export $(REPORT_HTML)`
- **Action**: Use `$(REPORT_HTML)` consistently in any target generating reports.

---

8. Environment & UID Files

- Files like `postman_env_uid.txt` and `postman_collection_uid.txt` are already variables (`ENV_UID_FILE`, etc.), but check if **raw paths** appear anywhere else in the Makefile. Replace them with `$(ENV_UID_FILE)` etc.

Suggested Refactoring Order

1. **Start with NPM/NPX commands** (immediate cleanup & DRY benefit).
2. **Move to jq scripts**.
3. **Check Postman path usage** (should be quick since you already have variables).
4. **Tighten Python environment references**.

Next Step

Would you like me to **rewrite your `install` target and give you 2–3 example refactored targets** (e.g., one that uses **`GENERATOR_OFFICIAL`**, one that uses **`PRISM`**), so you can see a clean pattern to follow?
”