PARITIES-1

This document enumerates intentional parity sets across the codebase: places that must remain "the same" in meaning and behavior. When any member of a parity set changes, update all other members accordingly.

Purpose: ensure future code AI agents automatically propagate consistent changes across paired components when modifying one of them.

Guidance for agents

- Prefer high-signal, obvious parities. Do not add speculative pairs.
- When adding a new feature/option, update all parity members in the same edit series.
- If you cannot satisfy a parity immediately, add a TODO in your PR description and open a follow-up issue.

Parity 1 — CLI options ↔ Context fields ↔ Defaults ↔ README

- Elements:
    - `src/prin/cli_common.py`: `parse_common_args(...)` option set and help
    - `src/prin/cli_common.py`: `Context` dataclass fields (comment: "Field list should match CLI options.")
    - `src/prin/defaults.py`: `DEFAULT_*` constants (values and choices, e.g., `DEFAULT_TAG_CHOICES`)
    - `README.md`: options and usage documentation (flags, behavior, examples)
- Invariant:
    - 1:1 mapping between CLI flags and `Context` fields, including default values from `defaults.py` and documented behavior in `README.md`.
    - Renaming/adding/removing a flag requires updating: parser, `Context`, `defaults.py`, and `README.md` in lockstep.

- Tests touching this parity:
    - Filesystem options: `tests/test_options_fs.py`
    - Repository options: `tests/test_options_repo.py`

Parity 2 — Tag choices ↔ Formatter classes ↔ README examples

- Elements:
    - `src/prin/defaults.py`: `DEFAULT_TAG_CHOICES` (e.g., `["xml", "md"]`)
    - `src/prin/prin.py`: tag→formatter dispatch (`{"xml": XmlFormatter, "md": MarkdownFormatter}`)
    - `src/prin/formatters.py`: `XmlFormatter`, `MarkdownFormatter`, `HeaderFormatter`
    - `README.md`: output examples for XML and Markdown
- Invariant:
    - Tag choices and dispatch table must stay in sync. Adding a tag requires a `Formatter` implementation, dispatch entry, defaults update, doc examples, and tests.
- Tests touching this parity:
    - `tests/test_options_fs.py::test_tag_md_outputs_markdown_format`
    - `tests/test_options_repo.py::test_repo_tag_md_outputs_markdown_format`

Parity 3 — --only-headers flag ↔ HeaderFormatter enforcement

- Elements:
    - `Context.only_headers` (CLI: `-l/--only-headers`)
    - `DepthFirstPrinter.__init__` forcing `HeaderFormatter` when `only_headers=True`
- Invariant:
    - When `only_headers` is set, bodies must not be printed regardless of the passed formatter.

- Tests touching this parity:
    - FS: `tests/test_options_fs.py::test_only_headers_prints_headers_only`
    - Repo: `tests/test_options_repo.py::test_repo_only_headers_prints_headers_only`

Parity 4 — Default filter categories ↔ Defaults ↔ README ↔ FS fixture

- Elements:
    - `src/prin/defaults.py`: `DEFAULT_EXCLUSIONS`, `DEFAULT_TEST_EXCLUSIONS`, `DEFAULT_LOCK_EXCLUSIONS`, `DEFAULT_BINARY_EXCLUSIONS`, `DEFAULT_DOC_EXTENSIONS`, `Hidden`
    - `README.md`: "Sane Defaults for LLM Input" section (categories listed)
    - FS test fixture tree: `tests/conftest.py::fs_root` (contains examples for each category)
- Invariant:
    - Categories defined in in `defaults.py` must be described in README, and corresponding sample files must exist in `fs_root` for coverage. If a category is added/removed/changed, update all three.
- Tests touching this parity:
    - FS flags toggling categories: `tests/test_options_fs.py` (e.g., `--hidden`, `--include-tests`, `--include-lock`, `--include-binary`, `--no-docs`, `--include-empty`, `--exclude`, `--no-exclude`, `--extension`)
    - Repo flags for analogous categories: `tests/test_options_repo.py`

Parity 5 — Exclusion/matching semantics shared across sources

- Elements:
    - `src/prin/core.py`: `DepthFirstPrinter._excluded` and `_extension_match`

  - `src/prin/filters.py`: `is_excluded`, `is_glob`, `get_gitignore_exclusions`
  - All adapters via `DepthFirstPrinter`: FS, GitHub, Website
- Invariant:
  - Inclusion/exclusion and extension matching must behave the same regardless of source. Any change to `filters` or engine matching must be validated against FS and Repo tests.
- Tests touching this parity:
  - FS: `tests/test_options_fs.py::test_exclude_glob_and_literal`, `::test_extension_filters_by_extension`
  - Repo: `tests/test_options_repo.py::test_repo_exclude_glob_and_literal`, `::test_repo_extension_filters`

Parity 6 — SourceAdapter protocol implemented uniformly by all adapters

- Elements:
  - Protocol: `src/prin/core.py`: `SourceAdapter` with `resolve_root`, `list_dir`, `read_file_bytes`, `is_empty`
  - Implementations: `src/prin/adapters/filesystem.py`, `src/prin/adapters/github.py`, `src/prin/adapters/website.py`
- Invariant:
  - Each adapter must implement the four methods with identical semantics expected by the engine:
    - `list_dir` raises `NotADirectoryError` when given a file path so explicit roots force-include
    - `resolve_root` returns a stable POSIX-like root for display-path calculations
    - `is_empty` semantics are consistent (see Parity 7)
- Tests touching this parity:
  - FS engine traversal/roots: `tests/test_cli_engine_tmp_path.py`, `tests/test_cli_engine_positional.py`

    - Repo positional semantics: `tests/test_print_repo_positional.py`
    - Mixed invocation: `tests/test_print_mixed_fs_repo.py`

Parity 7 — Semantic emptiness detection shared across adapters

- Elements:
    - `src/prin/core.py`: `is_blob_semantically_empty`, `_is_text_semantically_empty`
    - Adapters: FS (`is_empty` uses shared function), GitHub (`is_empty` uses shared function), Website (`is_empty` returns False; emptiness determined after download if needed)
- Invariant:
    - A single definition of "semantically empty" governs FS and GitHub sources (Python-only at present). Changes must update both adapters and associated tests.
- Tests touching this parity:
    - FS: `tests/test_filesystem_source.py` (empty/non-empty Python and text files)
    - Repo: `tests/test_options_repo.py::test_repo_include_empty`

Parity 8 — Display-path normalization across sources

- Elements:
    - `DepthFirstPrinter._display_path` and anchor-base logic in `run`
    - Adapter `resolve_root` implementations (FS yields absolute POSIX, GitHub uses repo-relative, Website uses a virtual root)
- Invariant:
    - Printed paths are relative to the provided roots (or the anchor base) and use POSIX separators consistently across sources.
- Tests touching this parity:
    - FS: `tests/test_cli_engine_positional.py`
    - Repo: `tests/test_print_repo_positional.py`

Parity 9 — Global file budget across sources (`--max-files`)

- Elements:
  - `src/prin/core.py`: `FileBudget`
  - `src/prin/prin.py`: single `FileBudget` instance shared across FS/Repo/Website runs
- Invariant:
  - The budget must be enforced globally across all sources in one invocation. New sources must consume from the same budget.
- Tests touching this parity:
  - FS: `tests/test_max_files_fs.py`
  - Repo: `tests/test_max_files_repo.py`

Parity 10 — GitHub URL subpath handling

- Elements:
  - `src/prin/util.py`: `extract_in_repo_subpath` (parses `/blob/`, optional branch, subpaths)
  - `src/prin/prin.py`: derives repo roots from the extracted subpath and sets `repo_ctx = ctx.replace(no_ignore=True, paths=[""])`
- Invariant:
  - URL-to-root translation logic in `util` and its use in `prin.main` must agree so that explicit file or subdirectory URLs behave as explicit roots and force-include as needed.
- Tests touching this parity:
  - Repo positional cases: `tests/test_print_repo_positional.py`

Parity 11 — CLI alias behavior

- Elements:
  - `src/prin/cli_common.py`: `CLI_OPTIONS_ALIASES` expansion (e.g., `-uu` → `--hidden --no-ignore`)

        - Direct short/long flags declared on the parser (e.g., `-u/--unrestricted`, `-uuu/--no-exclude`)
- Invariant:
    - Aliases must expand to semantically equivalent flag sets. Keep alias table, parser declarations, and README consistent.
- Tests touching this parity:
    - FS: `tests/test_options_fs.py::test_uu_includes_hidden_and_gitignored`, `::test_unrestricted_includes_gitignored` (note: `.gitignore` parsing is intentionally skipped)

Parity 12 — Test coverage parity for FS vs Repo

- Elements:
    - Options exercised in both suites: `tests/test_options_fs.py` and `tests/test_options_repo.py`
    - Budget tests: `tests/test_max_files_fs.py` and `tests/test_max_files_repo.py`
- Invariant:
    - Mature CLI behaviors should be covered for both adapters (filesystem and GitHub), unless the feature is intentionally source-specific. When adding a new option/behavior, add or adapt tests in both locations.

Parity 13 — Website adapter URL list parsing ↔ tests

- Elements:
    - `src/prin/adapters/website.py`: `_parse_llms_txt`, URL resolution, key naming/dedup logic
    - Tests: `tests/test_website_adapter.py`, `tests/test_website_adapter_all_urls.py` (monkeypatch `_parse_llms_txt` and assert all URLs are printed)
- Invariant:
    - The adapter's interpretation of `llms.txt` and header naming must

remain stable with the tests' expectations. Changes here require test updates.

Candidates to confirm (borderline parities)

- Filters classifier coupling: `filters.is_glob` delegates to `path_classifier._is_glob`; tests live in `tests/test_pattern_classifier.py`. If classification rules change, `is_excluded` behavior can shift. Treat as a soft parity between `path_classifier.py` and `filters.py`.

PARITIES-2

PARITIES

Purpose: Ensure that when any member of a parity set changes, all other members are reviewed/updated to maintain intentional 1:1 or N:N consistency. These are deliberate couplings, not smells. The threshold for inclusion is high and obvious.

Conventions

- Each set has: ID, Members (with exact locations), Contract (what must stay in lockstep), Triggers (what events require syncing), and How to Update (quick checklist/tests).
- "Members" list exact symbols or files; ranges are avoided. If a member is removed, either remove the set or replace the member accordingly.

SET: CLI-CTX-OPTIONS

- Members
  - README.md: Options documented under "Options Roadmap" and behavior narratives

    - src/prin/cli_common.py: `Context` fields and default values; `parse_common_args(...)` arguments and flags; `_expand_cli_aliases` map
    - src/prin/defaults.py: `DEFAULT_*` used by CLI defaults and choices
    - src/prin/core.py: `DepthFirstPrinter._set_from_context` consumed fields and runtime behavior tied to flags
    - tests/test_options_fs.py, tests/test_options_repo.py: end-to-end option coverage
- Contract
    - 1:1 parity between CLI flags, `Context` fields, and defaults in `defaults.py`. Adding/changing a flag requires adding/changing the matching `Context` field and default constant, and adjusting `DepthFirstPrinter` consumption when applicable.
    - CLI alias expansions must reflect the same semantic behavior as the canonical flags.
    - README must describe every implemented flag with correct semantics; planned flags must not be claimed implemented.
- Triggers
    - Adding/removing/renaming a CLI flag; changing a default; changing how a flag affects traversal, filtering, or output.
- How to Update
    a. Update `defaults.py` constants
    b. Update `Context` field list and `parse_common_args`
    c. Update `_expand_cli_aliases` if aliases change
    d. If behavior changes, update `DepthFirstPrinter._set_from_context` and friends
    e. Update README’s option documentation
    f. Extend/adjust tests in `tests/test_options_*.py`

SET: FORMATTER-SELECTION

- Members

    - src/prin/prin.py: selection of formatter by `ctx.tag` ("xml" → `XmlFormatter`, "md" → `MarkdownFormatter`)
    - src/prin/formatters.py: `XmlFormatter`, `MarkdownFormatter`, `HeaderFormatter` semantics
    - src/prin/core.py: `DepthFirstPrinter` forcing `HeaderFormatter` when `only_headers` is true
    - tests/test_options_fs.py::test_tag_md_outputs_markdown_format
    - tests/test_options_repo.py::test_repo_tag_md_outputs_markdown_format
- Contract
    - Tag strings available in CLI must have a matching formatter class and identical mapping in `prin.py` and `defaults.DEFAULT_TAG_CHOICES`.
    - `only_headers` forces header-only output regardless of selected formatter.
- Triggers
    - Adding a new tag value; changing behavior/format of a formatter.
- How to Update
    a. Add formatter class
    b. Add tag value to `DEFAULT_TAG_CHOICES` and mapping in `prin.py`
    c. Adjust tests to assert new format, keep header-only override behavior
    d. Update README

SET: SOURCE-ADAPTER-INTERFACE

- Members
    - src/prin/core.py: `SourceAdapter` Protocol and `Entry`/`NodeKind`
    - src/prin/adapters/filesystem.py: `FileSystemSource`
    - src/prin/adapters/github.py: `GitHubRepoSource`
    - src/prin/adapters/website.py: `WebsiteSource`
    - tests/test_filesystem_source.py, tests/test_github_adapter.py, tests/test_website_adapter.py, tests/test_website_adapter_all_urls.py

- Contract
  - All adapters implement: `resolve_root`, `list_dir`, `read_file_bytes`, `is_empty` with the semantics expected by `DepthFirstPrinter`:
    - `resolve_root` returns a logical POSIX path used for display anchoring
    - `list_dir` raises `NotADirectoryError` when the input refers to a single file (to force-include explicit paths)
    - `read_file_bytes` returns raw bytes
    - `is_empty` uses shared semantic emptiness for Python where applicable
  - Path display must be interoperable (POSIX-like) to keep formatting consistent across adapters.
- Triggers
  - Adding a new adapter; changing the protocol or `Entry`/`NodeKind` shapes or semantics.
- How to Update
  a. Update `SourceAdapter` Protocol and all adapters to match
  b. Ensure explicit-file behavior via `NotADirectoryError` parity
  c. Keep POSIX-style paths for display
  d. Add/adjust adapter-specific tests; ensure mixed-source tests still pass

---

SET: ENGINE-FILTERS-SEMANTIC-EMPTINESS

- Members
  - src/prin/core.py: filtering hooks (`_excluded`, `_extension_match`, `is_blob_semantically_empty`), budget handling, header-only behavior
  - src/prin/filters.py: `is_excluded`, `get_gitignore_exclusions`, `is_glob`, `is_extension`
  - src/prin/defaults.py: default exclusion sets and categories (tests, lock, binary, docs, hidden)
  - tests/test_cli_engine*.py, tests/test_options*.py

- Contract
    - Engine filter behavior must reflect CLI context and defaults consistently; explicit positional paths are force-included regardless of exclusions.
    - Semantic emptiness for Python is shared and adapter-agnostic; toggled by `include_empty`.
    - `--max-files` applies globally across sources via `FileBudget`.
- Triggers
    - Changing filter semantics, default exclusion sets, or emptiness logic.
- How to Update
    a. Adjust `defaults.py` and `filters.py`
    b. Ensure `Context.__post_init__` composes final exclusions correctly
    c. Verify engine respects force-include and budget semantics
    d. Update README behavior narratives and tests

SET: CLI-URL-ROUTING

- Members
    - src/prin/prin.py: input token routing between filesystem, GitHub, and website URLs; repo subpath extraction; global `FileBudget`
    - src/prin/util.py: `is_github_url`, `is_http_url`, `extract_in_repo_subpath`
    - tests/test_print_repo_positional.py, tests/test_print_mixed_fs_repo.py, tests/test_max_files_*.
- Contract
    - Routing logic and helpers remain in lockstep: a token classified as GitHub must be handled by GitHub adapter; HTTP non-GitHub goes to Website adapter; everything else local filesystem. Subpath extraction must be reflected in traversal roots.
- Triggers
    - Changing URL detection or subpath rules; adding new source kinds.
- How to Update

    a. Update helpers in `util.py`
    b. Update routing in `prin.py`
    c. Extend tests to cover mixed inputs and edge cases
    d. Update README examples

SET: PATTERN-CLASSIFIER

- Members
    - src/prin/path_classifier.py: `classify_pattern` and `_is_glob`
    - src/prin/filters.py: `is_glob` re-export and use
    - tests/test_pattern_classifier.py
    - src/prin/**init**.py: re-export for external tests
- Contract
    - Classifier rules must be consistently used by filters; re-exports must remain aligned with tests' import paths.
- Triggers
    - Changing classifier heuristics or moving exports.
- How to Update
    a. Update classifier
    b. Keep `filters.is_glob` parity
    c. Adjust tests and any re-exports

SET: README-EXAMPLES-REALITY

- Members
    - README.md examples and documented behavior
    - src/prin/prin.py and adapters for actual observed behavior
    - tests that cover the same stories (options tests, mixed-source tests)
- Contract
    - README claims must reflect implemented behavior and flags; examples should run as described.

- Triggers
    - Any behavior or flag change; example updates.
- How to Update
    a. Update README text and examples
    b. Ensure corresponding tests still pass

SET: TEST-SUITE-COVERAGE-BY-FEATURE

- Members
    - tests/test_options_fs.py, tests/test_options_repo.py: cover each CLI flag end-to-end per source
    - tests/test_cli_engine_*.py: traversal and path display behavior
    - tests/test_max_files_*.py: `--max-files` budget semantics
    - tests/test_website_adapter_*.py: website parsing and rendering
- Contract
    - For each implemented feature/flag, there is test coverage for both local filesystem and GitHub sources when applicable; website adapter covered for its specific behavior.
- Triggers
    - Adding a new CLI flag or behavior; adding a new adapter.
- How to Update
    a. Add parallel tests for each source where feature applies
    b. Keep assertions aligned (differ only by adapter-specific expectations)

SET: EXPLICIT-PATH-FORCE-INCLUDE

- Members
    - src/prin/core.py: DFS handling of `NotADirectoryError` → force include; duplicate suppression
    - src/prin/adapters/github.py: file-path responses raise `NotADirectoryError`

    - src/prin/adapters/filesystem.py: `list_dir` uses scandir semantics; explicit file roots handled by engine
    - tests/test_cli_engine_positional.py::test_directory_and_explicit_ignored_file_inside
    - tests/test_print_repo_positional.py::test_repo_explicit_ignored_file_is_printed
- Contract
    - Passing an explicit path must print it even if default exclusions would skip it; applies uniformly across adapters.
- Triggers
    - Changing how explicit paths are routed or how adapters signal file vs directory.
- How to Update
    a. Ensure adapters raise `NotADirectoryError` for explicit file-path roots
    b. Keep engine's force-include behavior intact
    c. Verify tests for both FS and GitHub

SET: BUDGET-GLOBALITY

- Members
    - src/prin/core.py: `FileBudget`
    - src/prin/prin.py: single shared budget across all sources
    - tests/test_max_files_fs.py, tests/test_max_files_repo.py, tests/test_print_mixed_fs_repo.py
- Contract
    - One global budget is shared across all sources in a single invocation; stopping traversal when spent.
- Triggers
    - Changing how file limits apply or introducing per-source budgets.
- How to Update
    a. Keep `FileBudget` logic and shared usage consistent

b. Adjust tests to new semantics

SET: GITIGNORE-BEHAVIOR

- Members
    - src/prin/filters.py: `get_gitignore_exclusions` (currently returns [])
    - src/prin/cli_common.py: `Context.__post_init__` composition of exclusions with `no_ignore`
    - tests/test_options_fs.py::test_unrestricted_includes_gitignored (and skipped tests around no-ignore)
- Contract
    - Until implemented, `.gitignore` is effectively ignored unless behavior changes; flags (`--no-ignore`, `-u`, `-uu`) must remain consistent with current semantics.
- Triggers
    - Implementing real gitignore parsing.
- How to Update
    a. Implement `get_gitignore_exclusions`
    b. Revisit flag interactions in `Context.__post_init__`
    c. Unskip and/or add tests documenting the new behavior

SET: WEBSITE-LLMS-TXT-PARSING

- Members
    - src/prin/adapters/website.py: `_parse_llms_txt`, URL normalization, key deduplication
    - tests/test_website_adapter.py, tests/test_website_adapter_all_urls.py
    - src/prin/prin.py: website routing and `WebsiteSource` usage
- Contract
    - All Markdown links and raw URLs in llms.txt are parsed and fetched; duplicates deduped by key with suffixing; printed with selected formatter.

- Triggers
    - Changing llms.txt parsing or keying rules.
- How to Update
    a. Update parser behavior and key mapping
    b. Adjust tests for expected headers and content

Notes on non-parities (intentionally excluded)

- Internal variable names and helper private-method shapes that do not affect the public CLI, adapter protocol, or documented behavior are not parity-bound.
- Performance choices (sorting strategy, traversal order beyond documented behavior) are not parity-bound unless tests assert specifics.

PARITIES-DIFF

Shared points (by item name)

- Parity 1 — CLI options ↔ Context fields ↔ Defaults ↔ README and SET: CLI-CTX-OPTIONS
    - 1:1 mapping between CLI flags and Context fields, defaults from defaults.py, and README; aliases must be semantically equivalent; tests in options_fs/options_repo cover behavior.
- Parity 2 — Tag choices ↔ Formatter classes ↔ README examples and SET: FORMATTER-SELECTION
    - Tag values align with formatter classes and DEFAULT_TAG_CHOICES; README examples reflect output; tests assert Markdown selection.
- Parity 3 — --only-headers flag ↔ HeaderFormatter enforcement and SET: FORMATTER-SELECTION
    - only_headers forces header-only output regardless of selected formatter.
- Parity 6 — SourceAdapter protocol implemented uniformly by all adapters and SET: SOURCE-ADAPTER-INTERFACE

    - All adapters implement resolve_root, list_dir (raise NotADirectoryError for files), read_file_bytes, is_empty with consistent semantics; POSIX-like display paths for anchoring.
- Parity 7 — Semantic emptiness detection shared across adapters and SET: ENGINE-FILTERS-SEMANTIC-EMPTINESS (also noted under SOURCE-ADAPTER-INTERFACE)
    - Single definition for semantic emptiness (Python), used by adapters; include_empty toggles; tests cover FS/Repo.
- Parity 4 — Default filter categories ↔ Defaults ↔ README ↔ FS fixture and SET: ENGINE-FILTERS-SEMANTIC-EMPTINESS
    - Default exclusion categories and filter behavior reflect CLI context/defaults and README. (Doc 1 also ties this to FS fixture; see uniqueness below.)
- Parity 5 — Exclusion/matching semantics shared across sources and SET: ENGINE-FILTERS-SEMANTIC-EMPTINESS plus SET: PATTERN-CLASSIFIER
    - Inclusion/exclusion and extension matching behave the same across sources; classifier rules used by filters.
- Parity 8 — Display-path normalization across sources and SET: SOURCE-ADAPTER-INTERFACE
    - Printed paths relative to roots/anchor base with POSIX separators consistently across sources.
- Parity 9 — Global file budget across sources (`--max-files`) and SET: BUDGET-GLOBALITY (also referenced in SET: CLI-URL-ROUTING)
    - One global FileBudget shared across all sources in a single invocation.
- Parity 10 — GitHub URL subpath handling and SET: CLI-URL-ROUTING
    - Subpath extraction stays in sync between util and prin routing so explicit file/subdir URLs act as explicit roots.
- Parity 11 — CLI alias behavior and SET: CLI-CTX-OPTIONS
    - Aliases expand to semantically equivalent flags; keep alias table, parser declarations, and README aligned.
- Parity 12 — Test coverage parity for FS vs Repo and SET: TEST-SUITE-COVERAGE-BY-FEATURE
    - Mature CLI behaviors have parallel coverage for filesystem and GitHub

(and website where applicable).

- Parity 13 — Website adapter URL list parsing ↔ tests and SET: WEBSITE-LLMS-TXT-PARSING
    - llms.txt parsing, URL normalization and deduplication, and tests' expectations remain aligned.
- Gitignore behavior (note)
    - Parity 11 tests note that .gitignore parsing is intentionally skipped; SET: GITIGNORE-BEHAVIOR codifies current semantics (get_gitignore_exclusions returns [], flags remain consistent; tests reflect this).
- README conformance (note)
    - Multiple Parities in doc 1 require README alignment and doc examples; SET: README-EXAMPLES-REALITY asserts README claims must match implemented behavior and runnable examples.
- Explicit-path force-include (note)
    - Parity 6 requires adapters to signal files via NotADirectoryError so explicit roots are force-included; SET: EXPLICIT-PATH-FORCE-INCLUDE makes this a dedicated contract with tests.

Unique to the first text (PARITIES-1.md)

- Parity 4 binds default filter categories to a specific FS test fixture: tests/conftest.py::fs_root must contain examples for each category. // ← keep
- Parity 10 calls out a specific implementation detail in prin.py when handling GitHub URL subpaths: sets repo_ctx = ctx.replace(no_ignore=True, paths=[""]). // ← discard
- Candidates to confirm (borderline parities)
    - Filters classifier coupling between filters.is_glob and path_classifier._is_glob treated as a soft parity. // ← discard
- Change checklist: a single global checklist to apply when any parity is affected.

Unique to the second text (PARITIES-2.md)

- Conventions section: standardized structure for sets (ID, Members, Contract, Triggers, How to Update).

- SET: CLI-URL-ROUTING covers general token routing across filesystem, GitHub, and generic HTTP (Website) using util.is_github_url and util.is_http_url; doc 1 only covers the GitHub subpath case in Parity 10.
- SET: PATTERN-CLASSIFIER elevates classifier coupling to a first-class parity (doc 1 treats it as a soft candidate).
- SET: README-EXAMPLES-REALITY is a dedicated parity ensuring README examples and claims match behavior.
- Notes on non-parities: explicit list of intentionally excluded concerns (e.g., private helper shapes, performance specifics).

Contradictions

- None found. The texts are consistent; the second text generally broadens or formalizes points present in the first.