Quickstart Flags & Modes — DeFi + ARC (Tier 1 & Tier 2)

This is the single, living reference for how to drive **micro-lm** via the **quickstart** entry point (CLI + Python). It unifies Tier-1 and Tier-2 semantics across **DeFi** and **ARC** and explains how flags interact.

---

0) TL;DR presets

Tier-1 (mapper audit only, no WDD)

```
python3 -m micro_lm.cli.defi_quickstart "swap 2 ETH for USDC"
  --rails stage11
  --policy '{"mapper":{"model_path":".artifacts/
defi_mapper.joblib","confidence_threshold":0.70}}'
  --verbose
```

Tier-2 (WDD detector audit; mapper decides action)

```
python3 -m micro_lm.cli.defi_quickstart "deposit 10 ETH into aave"
  --rails stage11
  --policy '{"audit":{"backend":"wdd"},"mapper":
{"confidence_threshold":-1.0}}'
  --verbose
```

Tier-2 (WDD family; WDD supplies plan)

```
python3 -m micro_lm.cli.defi_quickstart "swap 2 ETH for USDC"
  --rails stage11
  --policy '{"audit":{"backend":"wdd","mode":"family","K":12,"template_width":
64,"z_abs":0.55,"keep_frac":0.70}}'
  --verbose
```

Edge block (example: DeFi LTV/HF/oracle)

```
python3 -m micro_lm.cli.defi_quickstart "withdraw 5 ETH"
  --rails stage11
  --policy '{"ltv_max":0.60,"mapper":{"confidence_threshold":-1.0}}'
  --context '{"oracle":{"age_sec":5,"max_age_sec":30},"risk":{"hf":1.15}}'
  --verbose
```

---

1) The single entry point

- CLI: `python3 -m micro_lm.cli.<domain>_quickstart "<prompt>" [--rails <stage>] [--policy JSON] [--context JSON] [--verbose]`
- Python: `from micro_lm.cli.<domain>_quickstart import quickstart` → `quickstart(prompt, policy=None, context=None, rails="stage11", T=180)`

Uniform output contract (both domains):

```
{
  "prompt": str,
  "domain": "defi" | "arc",
  "rails": "stage11",
  "T": int,
  "top1": str|null,
  "sequence": [str],
  "plan": {"sequence": [str]},
  "verify": {"ok": bool, "reason": str, "tags": [str]},
  "flags": { ... },
  "aux": { "stage11": { "wdd": {...} }, ... },
  "wdd_summary": { "decision": "PASS"|"ABSTAIN"|null, "keep": [...], "sigma":
int|null, "proto_w": int|null, "which_prior": str|null, "note": str|null },
  "det_hash": str,
  "abstained": bool
}
```

Key invariants: - `plan.sequence` emits **canonical primitives** (e.g., `swap_asset`, `deposit_asset`). - If `verify.ok == false` (or reason contains a blocking keyword), **plan is cleared**: `plan.sequence = []`.

---

2) Policy schema (flags)

2.1 `policy.mapper` (Tier-1 audit; “action source” in all tiers)

Key	Type	Default	Meaning
`model_path`	str	null	Path to trained mapper (joblib).

Key	Type	Default	Meaning
`confidence_threshold`	float	0.70	Tier-1 **audit gate**; if score < τ, mapper abstains. For Tier-2 runs, set to `-1.0` or use `skip_gate` to disable.
`skip_gate`	bool	false	(Optional toggle) If `true`, ignores `confidence_threshold` entirely. Useful for Tier-2 WDD.

Notes - The mapper is the **authoritative action chooser**; we always canonicalize its label to a primitive for `plan.sequence`. - For **Tier-2**, disable the gate (`confidence_threshold:-1.0` or `skip_gate:true`) so WDD is the audit.

2.2 `policy.audit` (Tier-2 audit)

Key	Type	Default	Meaning
`backend`	enum	null	Set to `"wdd"` to enable WDD.
`mode`	enum	`"detector"`	`"detector"` (audit only) or `"family"` (WDD returns `order` as plan).
`gate`	bool	false	If `true`, overwrite `verify.ok/reason` with WDD's decision (rare; detector is usually non-authoritative).
`K`	int	12	Family sampler size.
`template_width`	int	64	Family template width.
`z_abs`	float	0.55	Family z-score threshold.
`keep_frac`	float	0.70	Family keep fraction.
`overrides`	dict	{}	Family parameter overrides per prior.
`pca_prior`	str	null	Optional `.npz` for WDD PCA prior.
`debug`	bool	false	Extra debug in `aux.stage11.wdd.debug`; enable `MICRO_LM_WDD_DEBUG=1` to print.

Detector behavior: - WDD attaches to `aux.stage11.wdd`, and `wdd_summary` mirrors essential fields. - `verify.reason` remains rails/local unless you set `audit.gate: true`.

Family behavior: - Short-circuits: returns `verify.reason = "wdd:family:<domain>"`, `flags.wdd_family = true`, and `plan.sequence = aux.wdd.order`.

2.3 Domain policy knobs

DeFi (examples): | Key | Type | Meaning | |---|---|---| | `ltv_max` | float | Max loan-to-value allowed. | | `hf_min` | float | Min health factor. | | `oracle.max_age_sec` | int | Price freshness requirement (in `context`). |

ARC (examples; align to your research checks): | Key | Type | Meaning | |---|---|---| | `must_cite` | bool | Block if no citation coverage. | | `rag.max_age_sec` | int | Retrieval freshness constraint. |

---

3) Environment variables

Var	Effect
`MICRO_LM_WDD_DEBUG=1`	Prints WDD detector debug lines (act, layer, sigma, prior, MF_peak).
`MICRO_LM_BENCHMARK_PLAN_OFF=1`	(Optional) Suppress `plan` for legacy benchmarks that only read `label`.

---

4) Semantics: who decides what?

1) **Mapper decides the action** (always). We canonicalize to `*_asset` (DeFi) or the ARC primitive set. 2) **WDD audits** the mapper's action. - Detector: advisory; metrics in `aux.stage11.wdd` + `wdd_summary`. - Family: authoritative planner when requested (policy `mode:"family"`). 3) **Verify = rails AND local domain verify**. - If local verify blocks (e.g., `ltv`, `hf`, `oracle` in DeFi), we **clear the plan** and set `verify.ok=false` with the domain reason.

`verify.reason` **cheat-sheet** - Rails shim only: `"shim:accept:stage-4"` (non-blocking default). - Detector annotated (optional): `"shim:accept:stage-4; wdd:pass"`. - Family: `"wdd:family:<domain>"`. - Local block (DeFi): contains `"ltv"`, `"hf"`, or `"oracle"`; plan is empty.

---

5) CLI patterns & gotchas

- **Quote JSON** with single quotes in zsh/macOS.
- `--use_wdd` (if present) is shorthand for `--policy '{"audit":{"backend":"wdd"}}'`.
- Disable Tier-1 gate in Tier-2 runs via `"mapper":{"confidence_threshold":-1.0}` or `"skip_gate":true`.
- Family mode parameters (`K`, `template_width`, `z_abs`, `keep_frac`) only apply in `mode:"family"`.

---

6) FAQ

Q: Why do I still see `shim:accept:stage-4` when WDD is on?
A: Detector mode is advisory by default. See `policy.audit.gate:true` to make WDD authoritative for `verify`.

Q: How do I know WDD fired?
A: `verify.tags` includes `audit:wdd` (detector), and `aux.stage11.wdd` + `wdd_summary` are populated. With `MICRO_LM_WDD_DEBUG=1` you'll see `[WDD]` lines.

Q: Mapper vs `_infer_action`?
A: WDD now prefers the **mapper-provided** canonical sequence; regex `_infer_action` is only a fallback if no sequence is given.

Q: What clears the plan?
A: Any `verify.ok=false` (e.g., `ltv/hf/oracle`), or explicit block reasons in the domain verifier.

---

7) Benchmark via quickstart (single surface)

Example DeFi Stage-8 run:

```
PYTHONWARNINGS="ignore::FutureWarning"
python3 scripts/tier2_benchmark.py
  --domains defi --runs 1
  --policy '{"mapper":{"model_path":".artifacts/
defi_mapper.joblib","confidence_threshold":0.35},"ltv_max":0.75,"hf_min":
1.0}'
  --outdir .artifacts
```

For Tier-2 WDD audits during the bench, disable the Tier-1 gate and add `audit.backend`:

```
PYTHONWARNINGS="ignore::FutureWarning"
python3 scripts/tier2_benchmark.py
  --domains defi --runs 1
  --policy '{"audit":{"backend":"wdd"},"mapper":
{"confidence_threshold":-1.0},"ltv_max":0.75,"hf_min":1.0}'
  --outdir .artifacts
```

---

8) Appendix — canonical primitive maps

DeFi

```
swap          → swap_asset
deposit       → deposit_asset
withdraw      → withdraw_asset
stake         → stake_asset
unstake       → unstake_asset
borrow        → borrow_asset
repay         → repay_asset
claim_rewards → claim_rewards_asset
```

ARC (example; align to your templates)

```
classify      → classify_text
extract       → extract_spans
summarize     → summarize_doc
qa|rag        → retrieve_and_answer
```

> Keep this doc open while we integrate ARC. We'll add ARC's concrete policy keys (e.g., citation coverage) and examples as we wire them.