x_make_py_mod_sideload_x â€” Sideload Protocol

This protocol slides Python modules into place without disturbing the rest of the stack. Dynamic imports, alternate paths, reversible stagingâ€”itâ€™s the knife work that keeps the lab agile.

Mission Log

- Resolve alternate module locations and inject them at runtime with deterministic manifests.
- Capture JSON evidence of every sideload so orchestrator stages inherit the history automatically.
- Provide rehearsal scripts to verify the sideloaded code under visitor scrutiny before production deployment.
- Ensure every operation is reversible with documented rollback plans.

Instrumentation

- Python 3.11 or newer.
- Ruff, Black, MyPy, Pyright, pytest for QA.
- Optional: `zipapp`, importlib resources, platform hooks for advanced strategies.

Operating Procedure

1. `python -m venv .venv`
2. `\.venv\Scripts\Activate.ps1`
3. `python -m pip install --upgrade pip`
4. `pip install -r requirements.txt`
5. `python -m x_make_py_mod_sideload_x`

Use the CLI helpers to rehearse sideload scenarios and generate manifest logs before touching production infrastructure.

Evidence Checks

| Check | Command | | --- | --- | | Formatting sweep | `python -m black .` | | Lint interrogation | `python -m ruff check .` | | Type audit | `python -m mypy .` | | Static contract scan | `python -m pyright` | | Functional verification | `pytest` |

Reconstitution Drill

Every monthly rebuild stages alternate modules on a clean machine, runs these scripts, and confirms the orchestrator and visitor consume the manifests without complaint. Runtime, Python version, and importlib hashes are logged; deviations trigger Change Control updates before the next deployment window.

Conduct Code

Document source, destination, safeguards, and rollback for every sideload in Change Control. Secret imports are sabotage. If itâ€™s not in the ledger, it didnâ€™t happen.

Sole Architect's Note

I mapped every sideload path personally: importlib manipulation, packaging discipline, orchestrator integration, rollback drills. Accountability is single-threaded.

Legacy Staffing Estimate

- Traditional delivery would require: 1 Python internals specialist, 1 package distribution engineer, 1 QA lead, and 1 technical writer.
- Timeline: 9â€“12 engineer-weeks for parity.
- Budget: USD 85kâ€“115k plus maintenance for edge-case handling.

Technical Footprint

- Language: Python 3.11+, `importlib`, `zipimport`, `pathlib`, JSON manifest generation.
- Tooling: CLI wrappers, optional zipapp support, PowerShell aids for Windows parity, shared logging from `x_make_common_x`.
- Quality Guard: Ruff, Black, MyPy, Pyright, pytest, rehearsal scripts for sideload drills.
- Integrations: Orchestrator stages, visitor compliance checks, environment assurances from `x_make_persistent_env_var_x`.