2026-04-29 Governance weekly

- Date: 2026-04-29

Conclusion / Action items

Flakes eval/lang restrictions

- piegames: let's rehash things to see if we agree or not, there has been conversations in the zulip
- piegames: basically, the main unresolved question is: do we feel strongly about those checks in the first place? how do we feel about changing them? how do we feel about removing them? from what i understood: we see those checks as mostly a guideline we can change over time but we want to keep them. one proposal that stuck out how to modify them would be to lift all the parser restrictions and go to only having a "no-function-calls" eval restriction.
  - Raito: full agreement on "from what I understood"
    - Hates those checks
    - However would be a huge pain in the ass to deal with users that are confused by the discrepancy of CppNix vs Lix
      - Because users often send us traces from not realizing it' CppNix instead of Lix
      - Unsolved problem of making users more sure of this
      - Predicts we get a lot of reports of "ahh everything is broken" because they try to eval CppNix-incompatable flakes not realizing they're using CppNix
  - Lunaphied: why do we want to remove the checks anyway?
  - raito: because in the bytecode world, there's no AST anymore.
  - qyriad: it would make error messages way better.
  - Qyriad: Good point on getting reports not realizing they're using CppNix. I would like though to consider how many people are going to make use of non-trivial flake.nix entrypoints and especially users that are not rea
  - Raito: two kinds of users of Nix:
    - Users of things like flake-parts
    - Knowledgeable about Nix; understand the two "languages"
    - They might exploit intentional features of Lix
    - People already do this; e.g. Lix can have a devshell with systemd; CppNix cannot
    - Other kind of users will fall into these errors
  - horrors: the delta between "no-function-calls" and "no weird AST" is thin and CppNix will probably follow this
  - piegames: if we are having an issue having breaking changes, i don't understand the difference between relaxing changes and removing them, it will cause the same kind

of incompatibility either way, it's a question of magnitude in a way, we will always have leaky abstractions; imagine someone doing a toString interpolation, with Lix now, we have `${32}` with `coerce-integers`, those details will always leak. I think the entire concept of restricting evaluation of expressively strong language is fundamentally the wrong approach, unless Flakes manage to go to JSON/TOML or something, those will always happen.
- raito: in full agreement with horrors
  - Also agree with piegames
  - The magnitude will be very low
  - Most users won't run into this
- raito: thinks we can merge Qyriad's original CL (Q: yay!)
- raito: envisions two outcomes:
  - People that want to replace npins/whatever
  - Granted without function calls this is pretty restricted
  - Taking a step back: we want some interface that works quickly, and flakes try to make that happen
    - This is a failure
    - The general idea, of enumerating metadata without long eval times (+ IFD, etc) is an interesting and valuable idea
- Qyriad: regarding the magnitude of concerns, it is low enough we can handle it, if it's not, then, i think we can put efforts to ensure that users can discover if they're not using lix or not; yes this idea of Flakes staying fast to enumerate is a complete failure, when it works, it barely works, the restriction are so much — most reasonable usecases don't even bother and go for legacy packages. The only thing that you can enumerate in Flakes without guaranteeing relatively short eval are the inputs and inputs alone. Everything else can take as long as it wants. I think if we want specifically make something that is quick to eval, we want to decide what are actual things we want to make it quick eval and put it in a TOML format.
- Raito: behind the current idea; just wanted to add some nuance
- piegames: lol i dont care

Summary / Action items

- Remove the AST-shape check
- forbid function calls (maxCallDepth=1)
- typecheck the evaluated result instead of checking the AST
- don't do anything about dyn-attrs (leave them as-is)
- notify the CppNix team

Discuss moving the testlib bits outside of the monorepo to address Python dependencies issues

- rootile: folks want to use F2 outside of the Lix codebase, what is the best way to facilitate that? snix used F2 as well; in my opinion, we might want to move testlib out of the F2 folder and make it a Python package which then you can use inside of test/f2 and even

publish it to PyPI so that people outside of Lix can use it too
- raito: someone could try to do an experiment of using f2 testlib outside the monorepo and show what it looks like and how it could be improved
  - For users of f2 you'd probably want both the Nix interpreter and the Python dependencies
  - Probably *wouldn't* publish it to Pypi
    - Since it depends on the Nix interpreter. Kinda weird
    - Then we have to maintain a publishing pipeline and stuff
  - If it's just a Nixlang entry point they can just put it in their build inputs and go
  - Wonders if the rest could be moved to async discussion
  - rootile: thinks that's fine

Trailing slashes in fetchurl (re: snix)

Context: https://git.lix.systems/lix-project/lix/issues/1185

- Difference between CppNix and Lix
- raito: What should the behavior be?
- horrors: we should change it; the current behavior makes no sense
- raito: :+1:

Quick check-in

- Raito: limited availablity
  - Flakes extraction is affected
    - Either need more help or move to next release cycle
  - Can still tackle merge queue
  - Also planning a LixCon retrospective

Revision #1
Created 2026-05-13 20:29:58 UTC by piegames
Updated 2026-05-13 20:32:55 UTC by piegames