Flood for Developers

There isno standard way to implement on-chain trading, developers must implement all execution logic themselves. They can then either come up with their own routing algorithm and execution logic, or sacrifice good execution for simplicity. Those brave enough to try the DYI route are soon met with a myriad of problems:

- How do you pick the best route?
- How do you split your trade across DEXs?
- How do you optimise your contracts on different chains?
- How do you protect from MEV?
- How do I monetise my orderflow in a sustainable way?
- How do you keep up and monitor your trading infrastructure?

Others swap on a single predetermined pool, choosing simplicity at the expense of good execution and enabling malicious front-running.

While traditional DEX aggregators solve a few of those issues for individual traders, they're not designed for application developers and thus have several shortcomings:

- You need their API to trade. Their uptime limits yours.
- You can get bad execution,sometimes, really bad(opens in a new tab)
- You have no control over who fills your trades, and end up selling your orderflow for cheap.
- Intent based aggregators are asynchronous and don't compose with synchronous smart contract logic.

If you would rather not deal with all of the above, Flood is for you. Flood provides excellent UX and execution withzero initial configuration , with full flexibility to add custom features when you need to.

Here's what Flood can do for you:

- Gasless trading for your users.
- Make your trading logic declarative rather than imperative.
- Execute smart contract logic atomically before and after a swap with hooks.
- Finegrained control over your orderflow with Zone Authentication. You choose who sees your orders, and who can fill them.

In Flood, developers are a first class citizen and Owners of their orderflow.

Last updated onFebruary 3, 2024 Offchain Infrastructure Writing your own Zone