SECTION 1: APPLICANT INFORMATION

Provide personal or organisational details, including the applicant's name, contact information, and the name of any associated organisation. This information ensures proper identification and communication throughout the grant process.

Applicant Name or Alias: [Enter full name]

zk50.eth

Project Name: [Enter Project Name]

GMX Rust SDK

Project Description: [Enter Project Description (1-3 sentences)]

Upon completion, this SDK will give users a performant, easy to use and fully open sourced set of functions for the GMX v2 contracts, written out fully in Rust using the ethers-rs and Alloy crates. Key functionality will include:

*A full read/write interface for the GMX v2 contract set, including hugely simplified methods for querying all relevant data (open interest, PnL(account), PnL(pool), etc.

Team Members and Qualifications: [List team members and their qualifications, roles, and responsibilities]

- zk50.eth - Rust/Ethereum dev.

Contact Information:

TG: [Telegram]

Telegram

https://twitter.com/50shadesofgwei_

Twitter: [Twitter Handle]

https://twitter.com/50shadesofgwei_

Email: [Email Address]

jonofeasby4@gmail.com

SECTION 2: GRANT INFORMATION

Detail the requested grant size, provide an overview of the budget breakdown, specify the funding and contract addresses, and describe any matching funds if relevant.

Requested Grant Size: [Enter Amount of ARB Requested]

80k ARB

Grant Matching: [Enter Amount of Matching Funds Provided - If Relevant]

n/a

Grant Breakdown: [Please provide a high-level overview and pro forma of the budget breakdown and planned use of funds]

The grant amount will be put towards paying one full time Rust developer to create a feature-complete SDK over the course of 2/3 months.

Funding Address: [Enter the specific address where funds will be sent for grant recipients]

After the proposal goes live, a 2/3 multisig will be created where 2 of the signers are known and respected members of the GMX community. (Call to Action: If this sounds like you, please reach out!)

Funding Address Characteristics: [Enter details on the status of the address; eligible address must be an MPC wallet or 2/3 multisig with private keys securely stored]

2/3 Multisig with 2 well trusted GMX community members.

Distribution Contract Address: [Enter any specific address that will be used to disburse funds for grant recipients]

n/a

Incentivised Contract Addresses: [Enter any specific contract addresses that will be incentivised in your program (Pool addresses, contracts, etc.)]

n/a

SECTION 3: GRANT OBJECTIVES AND EXECUTION

Clearly outline the primary objectives of the project and the Key Performance Indicators (KPIs) used to measure success. This helps reviewers understand what the project aims to achieve and how progress will be assessed.

Objectives: [Clearly state the primary objectives of the grant and what you intend to achieve]

Build a feature-complete, performant SDK that abstracts away the complexities of integration into the GMX protocol. The functionality will include both read methods (get open interest for a market, position info, etc), and write methods (open an arbitrary position, close an open position, provide/withdraw liquidity, etc)

Key Performance Indicators (KPIs): [Specify the KPIs, including but not limited to total value locked, transaction volume, and number of users that will be used to measure success in achieving the grant objectives]

- Number of forks
- Number of watchers/stars on Github
- The amount of volume routed to the UI fee address
- Feedback from users in a dev support TG channel, which will be created if/when the grant is cleared.

How will receiving a grant enable you to foster growth or innovation within the GMX ecosystem?: [Provide details]

I actually started this work a couple of weeks ago as the execution module of a personal trading bot project, and was immediately struck by the fact that the integration infrastructure was not yet in place; neither in Rust nor in any language. This should not be the case.

If on-chain finance is truly the future, it needs to outcompete the current centralised infrastructure, and at the minute it is simply far, far easier to integrate with CEX platforms than it is with their decentralised counterparts.

Furthermore, Rust as a programming language is very swiftly gaining popularity among Ethereum developers, MEV searchers and algo-traders for its performance and robustness (see: Arbitrum Stylus, RETH, Paradigm's Alloy/ethers-rs crates, and many others). Clearly, the industry is moving towards high-performing and reliable infrastructure, and GMX is in a prime position to take full advantage; high integration-time costs should not get in the way of the protocol seeing the fruits of this development, and and easy-to-use SDK made specifically to facilitate such high-value integrations is a necessary stepping stone towards that goal.

Justification for the size of the grant: [Enter explanation]

1 full-time Rust Engineer's salary, for around 2/3 months.

Execution Strategy: [*Describe the execution plan, including resources, products, use of funds, and risk management. This includes allocations for specific pools, eligible assets, products, etc.]

All of the features available on the GMX UI will be available through a crate of exported functions. The SDK's composition will be such that three guiding principles are adhered to throughout the entire codebase:

1. Abstract away as much complexity as possible. (Example: input vars for a market order are all of type String. Conversions to H160/U256 are handled on the backend, so a user doesn't have to worry about them).
2. The less steps it takes a new partner to integrate, the better.
3. All functionality should be fully documented, and detailed examples should be given for every method.

As noted before, the methods will include (but not be limited to):

- Monitoring key stats for GM pool composition, open interest, fast prices, available liquidity, funding rates, price impacts
- Opening and closing both standard market orders as well as complex orders (limit orders, TP/SL), performing swaps, providing and withdrawing liquidity

Upon completion of an alpha version, the SDK will be released as a crate and published to crates.io.

Grant Timeline & Milestones: [Describe the timeline for the grant]

Part 1 - Read functionality

Feature-complete read compatibility. Estimated time required: 2/3 weeks.

Part 2 - Write functionality

Feature-complete write compatibility. Estimated time required: 1.5/2 Months

Part 3 - Documentation

Fully write out examples and documentation for all of the including methods, including a step-by-step guide for how to install the SDK locally, a list of example environment variables, and a functioning example for each method.

Submit the documentation for review by the GMX team, and also to reputable Rust/Ethereum devs.

Fund Streaming: [Do you accept the funding of your grant streamed linearly for the duration of your grant proposal, and that the multisig holds the power to halt your streamed grant at their discretion at any time?]

Yes

SECTION 4: PROTOCOL DETAILS

Provide details about the protocol requirements relevant to the grant. This information ensures that the applicant is aligned with the technical specifications and commitments of the grant.

What date did you build on GMX?: [Date of deployment]

Started building a GMX interface around mid November, 2023.

Protocol Performance: [Detail the past performance of the protocol and relevance, including any key metrics or achievements, dashboards, etc.]

n/a

Protocol Roadmap: [Describe relevant roadmap details for your protocol or relevant products to your grant application.]

As detailed above, first the interface for the DataStore and Reader contracts will be created, including logic to abstract away key-hashing and syntax issues. After this, the logic for the ExchangeRouter/OrderVault contracts will be completed. (Note: Market increase/Decrease orders are already well underway and have functioning prototypes. Repo link at bottom of page.)

Audit History (if any): [Provide historic audits and audit results]

n/a

SECTION 5: Data and Reporting

Provide details on how your team is equipped to provide data and reporting on the grant distribution.

Is your team prepared to create Dune Spells and/or Dashboards for your incentive program?: [Please describe your answer]

It is possible to track usage via referral or volume routed to the UI fee address, or via the traction garnered on the Github Repo.

Does your team agree to provide bi-weekly program updates on the GMX Forum thread?: [Please describe your strategy and capabilities for data/reporting]

Yes, the repo and all relevant code is public and I highly encourage anyone interested to take a look and give feedback!

REPO URL: GitHub - 50shadesofgwei/GMXRust

ADDITIONAL NOTE: If you're an experienced Rust/Ethereum developer and want to be involved in the project, please don't hesitate to get in contact with me via any of the provided socials listed at the top of the proposal. I'm always eager to work with talented and passionate developers :).

Does your team acknowledge that failure to comply with any of the above requests can result in the halting of the program's funding stream?: [Y/N]

Y