Proposal: Maintenance for the substrate-api-client Nov22 - Jan23

Proponent: `FsnxqJnqWVNMZZgxaQdhaCk9c5sL3WSggRCRqp1qEzk1L2i`
Date: November 2022
Requested KSM: 2'866.6666 KSM
Short Description: This proposal aims to fund: 1. maintenance and support for the substrate api client over 3 months (Nov-22 until Jan-23). 2. evaluation and design of two new features: a CLI-wallet implementation and full no_std-compatibility

---

The substrate-api-client is a Rust-library for connecting to a substrate-based node via RPC. It is an alternative to subxt, which provides a similar functionality. RPC clients are needed in any software component, connecting to a substrate parachain. Therefore it is advisable, that there is more than one client available for Rust developers. Apart from that, the substrate-api-client fills a gap, providing the possibility to create extrinsics in a no_std-environment. Thereby facilitating extrinsic creation from within trusted hardware (like Intel sgx). This has a big potential for further usage in IoT environments. Last but not least, the substrate-api-client has an easy-to-use interface and comes with many practical examples, which makes it a good option for new developers.

Originally, substrate-api-client has been developed by SCS as a side product of a web3 foundation grant for SubstraTEE and has since been maintained and extended based on funds from both Encointer and Integritee. These teams have, however, only focused on the features they need for their own purposes and many good ideas and requests from the community have not been implemented yet.

The substrate-api-client is currently used by at least 5 ecosystem projects actively. There are **101 forks** on github and **over 20 Github organizations** currently have a dependency on the client. We therefore consider this library a common good for the ecosystem and we are looking for sustainable financing of enhancements but also maintenance to keep up with frequent substrate/polkadot releases.

Ongoing Maintenance

With this proposal we would like to fund ongoing maintenance for the existing functionality over the course of 3 months. This includes:

- Address current issues
- Stay up-to-date with the ecosystem
    - Follow Polkadot releases within a timely manner

    - Updates of used libraries (ws, serde, hex, …) as needed
    - Replacement of unmaintained libraries
- Address technical debt
    - Improvements on code-quality (refactorings, add more tests, …)
- Support existing and new users
    - Answer questions to usage and analyzing problems
    - Track current issues, document bugs and providing workarounds
    - Keep documentation up-to-date

Current Issues

As of November 2022, we would address the following list of issues:

- Ease-of-use
    - More robust error and timeout handling (#157, #134, #241)
- Improve performance to support more requests per second
    - Implement usage of concurrent threads (#133, #240).
    - If needed: change to different websocket library
- Improve documentation
    - Add more examples (#225)
    - Get documentation up-to-date (#277)

Long-term goals

Apart from the ongoing maintenance we want to work towards a broader usage of the api-client. In the following we describe our current long-term goals:

CLI-wallet implementation with decent security

Substrate-api-client has already been used to implement wallet functionality, but for test purposes only and without any security features. We see potential in developing a proper CLI wallet with 2FA. While such a cli wallet won't be as secure as a hardware wallet or air gapped parity signer phone, it will improve upon the security of the most popular wallet, the polkadot-js browser extension, which uses password protection only. We suggest using smartcard/yubikey features for enhanced protection of the keystore. The detailed design has yet to be elaborated and makes part of this proposal.

Currently there is no command-line wallet implementation (see https://wiki.polkadot.network/docs/build-wallets)

Tracking issues:

- https://github.com/scs/substrate-api-client/issues/266

Full `no_std` compatibility for IoT

`no_std`-compatibility is needed, whenever the std-library cannot be used for some reason. This is the case on some lightweight embedded devices, trusted execution environments (like Intel sgx) or from within a wasm blob. Currently, only part of the code is `no_std`-compatibile: extrinsics and metadata can be created and parsed. To actually send an extrinsic, via websockets, the std-library is still needed.
We propose to evaluate and work towards full `no_std`-compatibility, such that `no_std`-Rust code can communicate with an rpc-node.

This can greatly extend the scope of substrate towards IoT

Tracking issues

- https://github.com/scs/substrate-api-client/issues/279

Team

Supercomputing Systems AG (SCS), based in Zurich, Switzerland

As an engineering services company SCS AG has more than 25 years of experience in the fields of electronics, software and system design. Profound know-how, solid methodological competence as well as efficient project management are the foundation of our success. See our company website, with a description of our work.

In the last 2 years our team “decentralized systems” has been the main contributor for the Integritee core development, has contributed to Encointer and has been contracted by other ecosystem projects. Within those 2 years we developed from substrate beginners to an experienced team, knowing how to implement, design and maintain substrate projects in a challenging environment. With the development of Integritee core, the team brought substrate pallets into trusted execution environments, requiring a profound knowledge of the internals of substrate. Furthermore the team acquired a deep knowhow on no_std-development in Rust.

Our team on github:

- haerdib
- mullefel
- echevrier
- niederb

Costs

This proposal is offered at **2'866.6666 KSM**.

The following table gives an overview on the expected costs:

	Deliverables	**Cost**
Maintenance		
Current Issues	Fixing the issues defined in "Currrent Issues"	$ 30'667
Stay up-to-date with ecosystem	Updates from Nov-22 until Jan-23	$ 9'333
Technical debt	Code improvements	$ 5'333
User support	Support from Nov-22 until Jan-23	$ 6'667
Long term goals		
CLI-wallet	Analysis and Detail Design. No implementation planned in this phase.	$ 20'000
Full no_std compatibility for IoT	Analysis and Detail Design. No implementation planned in this phase.	$ 8'000
Total Costs: Maintenance and Long term goals		**2'750 KSM = $80'000**, based on EMA7 7 day moving average for the KSM/USD rate (29.088 USD/KSM)
Deposits Kusama		
Submission deposit for proposal		100 KSM
Deciding deposit for proposal		16.6666 KSM
Overall Total		**2'866.6666 KSM**

As the project serves a common good and will continue to be published under Apache 2 license, SCS offers a discount to their usual rate, resulting in a price of 165 CHF (Swiss Francs) per hour.

The costs of this proposal cover maintenance and support for 3 months plus an evaluation of our long-term goals. The idea is to continuously have follow-up proposals (for periods of 3 months), that will cover costs for maintenance and smaller work packages. For bigger work packages in the future (e. g. the implementation of a CLI-wallet) we will make separate proposals.

Maintenance will be done **from November 1st 2022 until January 31st 2023**. Analysis and Detail Design of the long term goals will also be done within this period.

At the end of each month, SCS will report on the issues they worked on. The progress can further be tracked in the github-repository.

In general, maintenance will be prioritized over long-term goals. Therefore the above plan is subject to change.

Beneficiary

`FsnxqJnqWVNMZZgxaQdhaCk9c5sL3WSggRCRqp1qEzk1L2i`

owned by:

Supercomputing System AG
8005 Zürich
Switzerland
CHE-107.465.659
E-mail: info@scs.ch