Light-Client Custom Token Creation Proposal

Why can’t we create custom tokens on top of Light-Clients

?

As it seems, the next evolution (as relating to Ethereum 2: POS Consensus) will be mobile-devices in regards to running the necessary nodes needed to maintain the ecosystem, and Light-Clients are the best arbiter to such a proposal.

As I’ve gathered, the best way to create custom tokens on top of Ethereum is to avoid dealing with Light-Clients

, but

, Light-Clients are built to interact with mobile phones (and other minimal devices) thus:

building the Infrastructure to create Custom Tokens & Tokenomics on top of Ethereum 2’s Light-Clients

…seems like the next best evolution to further expand Ethereum and its ecosystem.

The beautiful aspect of creating such custom tokens on top of Light-Clients is that such tokens can be created with multiple languages (Typescript, Go, Rust, Java, Nim, C#, etc) as opposed to learning a new language just to create an Ethereum token.

This I believe will onboard a plethora of new developers into the Ethereum ecosystem as the Research and Development needed to create such tokens in multiple languages has already commenced via the light-client building process:

Live Light-Clients:

- Typescript Light Client

(for Lodestar)

Needed Light-Clients:

- Go Light Client

(for Prysm) <~ (vacant link to-be-assigned)

- [Rust Light Client

(for Lighthouse)](https://github.com/jmcook1186/beacon-light-client) <~ (open for collaboration)

- [Java Light Client

(for Teku)](https://github.com/jeyakatsa/Altair----Minimal-Light-Client-Prototype) <~ (open for collaboration)

- Nim Light Client

(for Nimbus) <~ (vacant link to-be-assigned)

- C# Light Client

(for Cortex) <~ (vacant link to-be-assigned)

This will most likely be a multi-year pronged experiment & project and is ever-so evolving as more information is gathered.

I am very open to more thoughts, ideas and collaborations.