DAH/IBM Joint Project Proposal

We strongly believe that it is imperative that the Hyperledger Project quickly decides on a base on which to begin collaboration and innovation. We believe that the foundation of code should be capable of enabling the broad range of use cases that have been discussed and contemplated by the community.

It is clear that the community is split over use of the Bitcoin model (UTXO). We believe that the community should be looking to quickly realize a single implementation in which the UTXO model ***and*** alternate model(s) - that provide a more elegant (and likely better performing) approach to satisfying more complex use cases - are equally supported.

There are also both near and long-term objectives that we share: 1) for the Hyperledger project to have a viable, stable code base and 2) to have a ***single*** viable, stable code base..

Digital Asset's stack implements the Bitcoin protocol that is currently supported by Blockstream and others. DAH's stack also provides an easy to use API abstracting the lowlevel blockchain constructs into higher-level constructs that are more usable for specific use cases, such as the ones found in the financial industry.

However, the architecture and design of the IBM OBC lends itself to being more suitable as the foundation for longer-term goals of satisfying the broadest range of use cases.

What we propose is that we start with two initial code bases: 1) DAH Java + Blockstream's libconsensus and 2) IBM's OBC. In order that we can get ***quickly*** to a ***single*** stable, and viable code base, we offer the following two-phased approach:

- Ensure that code bases work as-is, and community effort should focus on (2) to quickly consolidate to 1 code base. This phase will ensure near term usability of the HLF code base, while incentivizing consolidation into phase 2.
- As per the previous IBM proposal, work towards extracting functionality from codebase (1) to be a specific instance of chain code and block validation of codebase (2), consolidate the clientserver API of the two codebases in order to allow the continued use of codebase (2) as the server for DAH higher level stack, delivering equivalent or better non-functional properties of performance and scale to the initial DAH code base.

We also propose that we would start with an initial week-long face-2-face sprint of the developers interested in helping to kick off the initial development as per the above proposal, and to help get the engineering community off to a good start with regards to team dynamics and shared understanding of the goals. We are currently seeking an available and suitable venue for the initial development sprint in NYC, to be conducted as soon as practicable.

Deliverables for initial sprint

Below is our analysis of concrete code deliverables to fill some of the conceptual gaps of OBC as relates to integrating support for the UTXO transaction model.

UTXO Transaction Object

Define a protobuf representation of a Transaction that unites requirements of OBC framework and that of UTXO. A transaction contains a list of referenced transaction outputs

and a list of new outputs. Both inputs and outputs may hold a DSL for execution by a chaincode.

Chaincode interpreting Transaction Script DSL

An implementation of the chaincode interface that

- uses the structural API to retrieve referenced transaction outputs
- concatenates referenced outputs DSL script with referencing input DSL script
- executes concatenated DSL script with our interpreter written in C++

Structural API Implementation

A stateless structural API implemented in Go and available both to chaincode (with scoped access) and to and higher order peer module, with following methods:

For data warehousing purposes:

- retrieve transaction by its unique identifier
- retrieve block by its unique identifier
- retrieve list of blocks IDs limited to a subset of the past
- retrieve transactions that contain a given data pattern in their serialization

For OLTP/Chain Code purposes:

- register a callback for new transactions
- register a callback for new blocks
- send a newly created transaction to network

Structural API to be implemented within OBC in Go that would connect an adapter that maps to the higher level stack in DAH Java implementation

The resulting architecture would look something like the following image: