**Hyperledger – Notes on the Hyperledger Fabric Composer project**

<https://hyperledger.github.io/composer/>

Chain code stub part: <https://github.com/hyperledger/composer/tree/master/packages/composer-runtime-hlf>

<https://hyperledger.github.io/composer/tutorials/developer-guide.html>

Composer is an application intended to build “business networks” to exchange assets.

To date, this is the only cross-asset prototyping effort I have seen on Hyperledger.

The architecture is very interesting:

* It is based essentially on a SDK application (nodejs)
* It deploys one chaincode for each “business network” (a set of participants, or “consortium”)
* The chaincode itself Is just a generic stub, running a nodejs engine
* Business logics implemented as JS code entered by an administrator as DATA on the chaincode (from the GUI)
* Business logics are therefore completely dynamic and “user driven”
* Business logics implements “digital assets” which may be exchanged
* One business network = one chaincode = 1 ledger for all participants of a single business network = registry of ownership for all assets owned by these participants

Pro:

* Business logics deployed from a GUI (flexibility)
* Cross-asset exchange made (relatively) easy
* One hard-time messing with Hyperledger / Chaincode intricacies, then switch to easier JS world
* Cool events notification features
* Powerful ACL feature (rules + JS)

Cons:

* Business logics as data: harder to certify smart contract behavior
* One place for all assets: delicate to deploy and validate new smart contracts independently
* Privacy: every node acting as a chaincode peer may inspect the whole registry (perhaps not from the GUI, but it is possible since data block are available locally an unencrypted)
* Model implies that transactions are made to exchange assets. Not that much flexible to adapt to other use cases
* Difficult to imagine how an “oracle” feature may be added to this framework

**Conclusion:**

* The idea of chaincode acting as a NodeJS wrapper (thanks to the DukTape package) is cool. However, I am not such a JS big fan, and GO makes do. What I like is the idea of a secured container deployed. If this container is just a stub, we open a big security hole.
* The idea to store JS smartcode logics as data is interesting, but I am afraid it is going to be difficult to validate business logics before deploying new “assets logics”
* The idea of a shared cross-asset ledger is obviously the only method available to exchange assets with Hyperledger (see my paper on data redundancy). However, privacy concerns are not addressed by this design. For our fund business, or for that matter, any use case in which participants are potential competitors, this is a show stopper.