Blockchain Integration Framework

Design Principles

1. Wide support – Interconnect as many ecosystems as possible regardless of technology limitations

2. Plugin Architecture from all possible aspects:

identities, DLTs, service discovery. Minimize how opinionated we are to really embrace interoperability rather than silos and lock-in. Closely monitor community feedback/PRs to determine points of contention where core BIF code could be lifted into plugins. Limit friction to adding future use cases and protocols.

3. Prevent Double spending Where Possible

Two representations of the same asset do not exist across the ecosystems at the same time unless clearly labelled as such [As of Oct 30 limited to specific combinations of DLTs; e.g. not yet possible with Fabric + Bitcoin]

4. DLT Feature Inclusivity

Each DLT has certain unique features that are partially or completely missing from other DLTs.
BIF - where possible - should be designed in a way so that these unique features are accessible even when interacting with a DLT through BIF. A good example of this

principle in practice would be Kubernetes CRDs and operators that allow the community to extend the Kubernetes core APIs in a reusable way.

5. Low impact

Interoperability does not redefine ecosystems but adapts to them. Governance, trust model and workflows are preserved in each ecosystem

Trust model and consensus must be a mandatory part of the protocol handshake so that any possible incompatibilities are revealed upfront and in a transparent way and both parties can “walk away” without unintended loss of assets/data.
The idea comes from how the traditional online payment processing APIs allow merchants to specify the acceptable level of guarantees before the transaction can be finalized (e.g. need a pin, signed receipt, etc).
Following the same logic, we shall allow transacting parties to specify what sort of consensus, transaction finality, they require.
Consensus requirements must support predicates, e.g. “I am on Fabric, but will accept Bitcoin so long as X number of blocks were confirmed post-transaction”
Requiring KYC (Know Your Customer) compliance could also be added to help foster adoption as much as possible.

6. Transparency

Cross-ecosystem transfer participants are made aware of the local and global implications of the transfer. Rejection and errors are communicated in a timely fashion to all participants.

Such transparency should be visible as trustworthy evidence.

7. Automated workflows

Logic exists in each ecosystem to enable complex interoperability use-cases.
Cross-ecosystem transfers can be automatically triggered in response to a previous one.

The automated procedure which is regarding error recovery and exception handling should be executed without any interruption.

9. Default to Highest Security

Protocol handshake must negotiate the highest common denominator of security.

10. Transaction Protocol Negotiation

Participants in the transaction must have a handshake mechanism where they agree on one of the supported protocols to use to execute the transaction. The algorithm looks for an intersection in the list of supported algorithms by the participants.

11. Participants can insist on a specific protocol by pretending that they only support said protocol only.

12. Protocols can be versioned as the specifications mature.

13. Adding new protocols must be possible as part of the plugin architecture allowing the community to propose, develop, test and release their own implementations at will.

14. The two initially supported protocols shall be the ones that can satisfy the requirements for Fujitsu's and Accenture's implementations respectively.

15. Means for establishing bi-directional communication channels through proxies/firewalls/NAT wherever possible.

Aspirational Principles (Will not translate into immediate roadmap but we hope to bring into roadmap eventually)

- Global identities – Include a way to identify the interoperable ecosystems, verify data provenance and build a reputation. Governance model, consensus algorithm, rules and participants are parts of an ecosystem identity [As of Oct 30 WIP e.g. Indy, but seeking enhancement on the communication protocol. Could provide a centralized mechanism or a chain of trust]
-

Some questions and points I'd like to see (Hartm):

1. Maybe this is a bit too early, but one thing I'd like to hear what you all think is how to handle componentization (i.e. how to set up the plugin architecture). What do you all think is the most natural division of modules? What sub-protocols or programs do you think we need? I'm curious to hear what everyone thinks.
2. I don't see much of anything in this document in the way of performance. Are there any performance requirements that you may have? In the long run, we would obviously want to be able to handle applications that require high performance. But, what kind of performance is required for your initial applications?