

Review on BIF whitepaper draft (on Use-cases, Interworking patterns)

Takuma TAKEUCHI
Fujitsu Laboratories Ltd.
<takeuchi.takuma@fujitsu.com>
RocketChatID: @takeutak

Feb 17, 2020

■ Overview

- Action item by next BIF phone meeting:
 - Make revision of the following draft of BIF whitepaper (written by Accenture)
<https://github.com/petermetz/blockchain-integration-framework/blob/docs/peter.somogyvari/whitepaper/docs/whitepaper/whitepaper.md>

■ Summary of our views

- "Interworking patterns" are needed to describe what interworking patterns BIF achieves (This should be reflected on our Hyperledger proposal)
 - Add section 4.1 "Interworking patterns"
- Moreover, the basic use case (section 2.1 "asset transfer") should be brushed up on some points.
 - Brush up section 2.1 "Asset transfer"
- Our review is reflected on a pull-request.

■ Our revision

■ Revise section 2.1 "Fabric to Quorum Asset Transfer" as the following:

2.1 Ethereum to Quorum Asset Transfer

Use Case Attribute Name	Use Case Attribute Value
Use Case Title	Ethereum to Quorum Asset Transfer
Use Case	<ol style="list-style-type: none"> User A owns an asset on a Ethereum ledger User A transfers some asset on Ethereum ledger to a Quorum ledger
Interworking patterns	Value transfer
Type of Social Interaction	Generic Asset Transfer
Narrative	A person (User A) has multiple accounts on different ledgers (Ethereum, Quorum) and they wish to transfer some asset from Ethereum ledger to a Quorum ledger. The asset they are transferring is a generic asset meaning that it doesn't have to be currency of any sort, but we assumed that User A agreed to release ownership of transferring asset on Ethereum ledger instead of getting ownership of transferred asset on Quorum ledger.
Actors	<ol style="list-style-type: none"> User A: The person or entity who has ownership of the transferred asset.
Goals of Actors	Some asset on Ethereum ledger had transferred to Quorum ledger.
Success Scenario	Transfer succeeds without issues. Asset is available on both Ethereum and Quorum ledgers.
Success Criteria	Presence of asset transfer across ledgers has cryptographic proof that is obtainable through BIF.
Failure Criteria	Asset transfer on both ledger is canceled.
Prerequisites	<ol style="list-style-type: none"> Ledgers are provisioned User A identity established on both ledgers. User A has access to BIF deployment
Comments	

Change Fabric to Ethereum (because Fabric doesn't have default assets)

Add the "Interworking patterns" column for showing the corresponding pattern (on section 4.1)

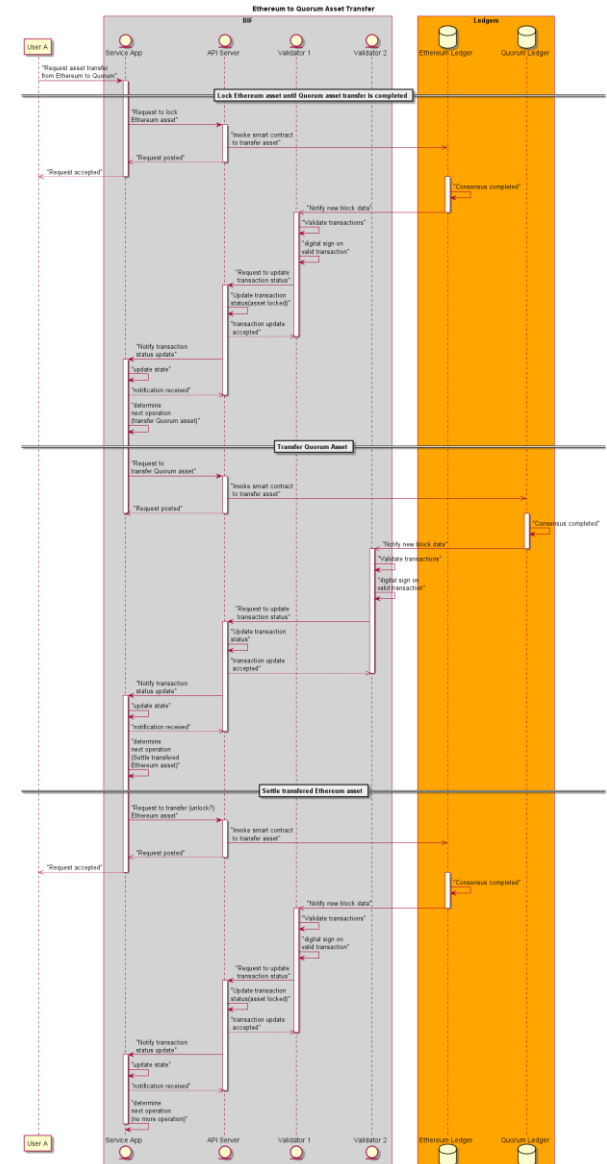
Moreover, some sentences are re-written.

The sequence diagram is re-designed (as the next page)

Our revision

Revise the sequence diagram of section 2.1

- The actor line "Service app" is added
- NOTE: Sorry that the right diagram is too small to see. Please refer the diagram on my pull-request.



■ Our revision

■ Add section 4.1 "Interworking patterns".

4.1.1 Interworking patterns list

The Blockchain Integration Framework has several interworking patterns as the following.

- Note: In the following description, **Value (V)** means numerical assets (e.g. money). **Data (D)** means non-numerical assets (e.g. ownership proof).
Ledger 1 is source ledger, Ledger 2 is destination ledger.

No.	Name	Pattern	Consistency
1.	value transfer	V -> V	check if V1 = V2 (as V1 is value on ledger 1, V2 is value on ledger 2)
2.	value-data transfer	V -> D	check if data transfer is successful when value is transferred
3.	data-value transfer	D -> V	check if value transfer is successful when data is transferred
4.	data transfer	D -> D	check if all D1 is copied on ledger 2 (as D1 is data on ledger 1, D2 is data on ledger 2)
5.	data merge	D <-> D	check if D1 = D2 as a result (as D1 is data on ledger 1, D2 is data on ledger 2)

■ Each patterns:

- "(1) Value transfer" is generalized from the use-case 2.1 "Fabric to Quorum asset transfer" and 2.4 "Stable Coin Pegged to Other Currency."
- "(2) Value-data transfer" and "(3) data-value transfer" are generalized from the use-case 2.2 "Escrowed Sale of Data for Coins."
- "(4) Data transfer" is generalized from the use-case 2.5 "Healthcare data sharing."
- "(5) Data merge" is generalized from the use-case 2.6 "Food traceability"

■ Our revision

■ Add section 4.2 .. 4.6 to describe the patterns on the above interworking list.

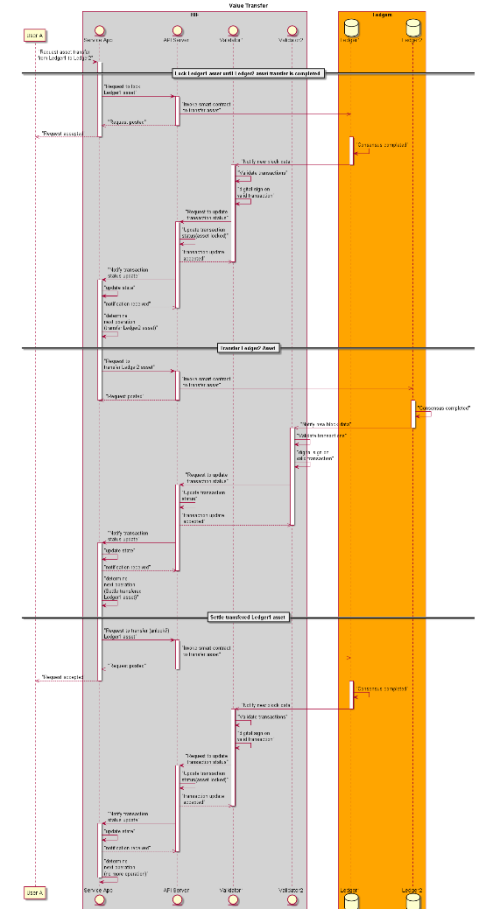
- 4.1.2 Value transfer
- 4.1.3 Value-data transfer
- 4.1.4 Data-value transfer
- 4.1.5 Data transfer
- 4.1.6 Data merge

■ Items for each section are as the following:

- Description of the pattern
- Sequence diagram

■ I suggest a sequence diagram for the basic pattern "Value Transfer" as the right diagram

- The sequence diagram is generalized from the one of section 2.1.
- NOTE: Sorry that the right diagram is too small to see. Please refer the diagram on my pull-request.



■ Current code on PlantUML (using `https://...` for viewing puml)

```
![Sequence Diagram - Fabric Quorum Asset Transfer](https://www.plantuml.com/plantuml/png/0/bLN1RXen4BtxAqRqqaf9AZYW4Blqg8IDHKall0Xunt0oiQcO-yc_7t6teKisuKstF1xysRcpGnSEI_9PqyfJF1FXaOXVD5oHV89pc4e5wGL_CeiPKymS0uzJ4aQjqGlmckV8dpVRf1IJ5P8S9rQzQZMKXiO6O-iuhKrC6GrVBRqE3FIJYALjmJt3ca-0Eb02h6mBz2w7WrRy66iLsl_ZUHMeE5ojj0Qt8su4ygHMf6-qMwFFSXK9vBr8fJpUm8iLYgEOM-kjQDpXlcDDPwuX7bP98UfTYEXB3yGEeAJo8LqMFOsBwG3cBJVm2Aa2dRUIfAPjwohS-aHh16YruD1vJSXjmfllHwJuBmVM-mUO8BGPJYsmLCxICMyMEUpixkmkUQo5y48I12WP8tuS2VFA2gtUMPvvmBTfuSumEJGqZ28Rz9So_UGkd2n8VUPkPZf5-4VSAavod3JsLisLl9lz-mdPZ0pn-Y2iQmUkOJu0ceDedbiZli2mz8Uhc5YLuPoU7ePiOkmrR0Vj6SKDmW3oMjq0euE5iANlgRyfAl2D7qxyWgv2AUjL_3rnTKMRYXoy-HqnUOykPKjDZpPZN2f3zOMF8G5PqxcvaAjb7EwGNVx-reUdk0IEybZurlU4EMQtBXW9B6XRmX3DyEJhUsKwSksb8TOJ2dRln-sVB_D3_ksea1L0KFXX1MM9iC8-UuvGtnUzkBK9Cqj694vMuw3NTJMd9mCfruCftToHHcrmx7ITijViCRj7ByBia_eMVHiKLDyp_9Qy0 "Sequence Diagram - Fabric Quorum Asset Transfer")
```

■ The style (using `https://..` for viewing puml) is inconvenient as the following reasons:


- On source codes, we cannot easily find the corresponding puml file on GitHub
- On off-line working environment, we cannot view the puml file

■ My suggested style

- Export puml file to png (or some image extension)
 - e.g.: VSCode package "PlantUML" has such a exporting function.
- On GitHub, we put both of puml and png.
- On markdown file, we write as the following:

```

```



FUJITSU

shaping tomorrow with you