mojaloop foundation

Mifos Payment Hub Updates

Enabling Upstream Innovation on an Open Payment Orchestration Engine

Edward Cable, The Mifos Initiative
Istvan Molnar, DPC Consulting

Agenda

- Introduction
- Payment Hub Origins
- Why It Matters - Business & Technical Benefits
- Payment Hub Enhancements
- Journey of Use Cases
  - Already Supported
  - New or In-Progress
- Roadmap
- Let's Collaborate!

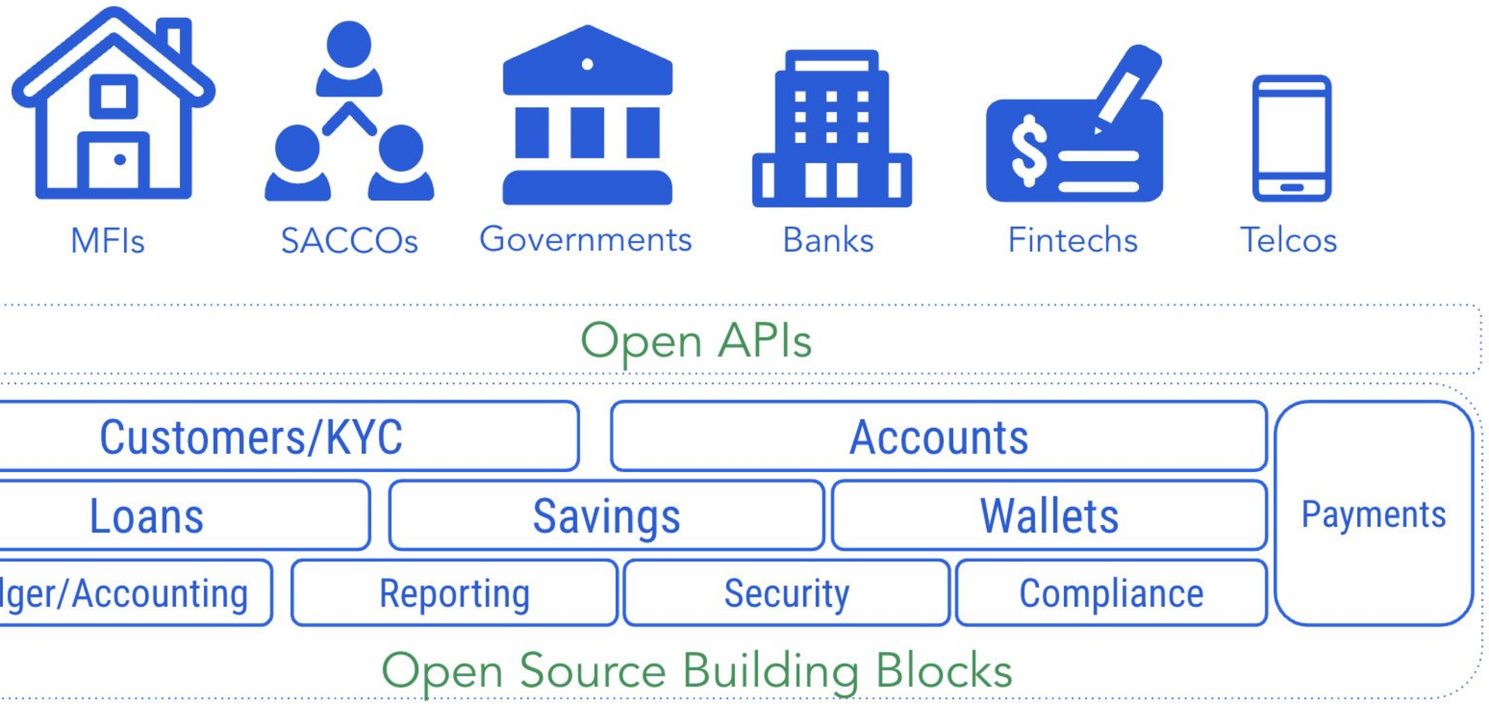
Open Source Building Blocks for Core Banking & Payment Orchestration

Modern, cost-effective highly connected composable banking infrastructure to:

- Advance financial health of 3B underbanked.
- Commoditize core banking software.
- Unlock virtuous cycle of upstream contribution & shared innovation.
- Enable global ecosystem of market-driven solutions & services.

20,000,000 clients reached by 500+ institutions across 56 countries powered by our open APIs.

- 2006. Launch at Grameen Foundation
- 2011. Independent 501(c)3
- 2017. Top-level Apache Project



Mifos Initiative

MIT SOLVE

Digital Public Goods Alliance

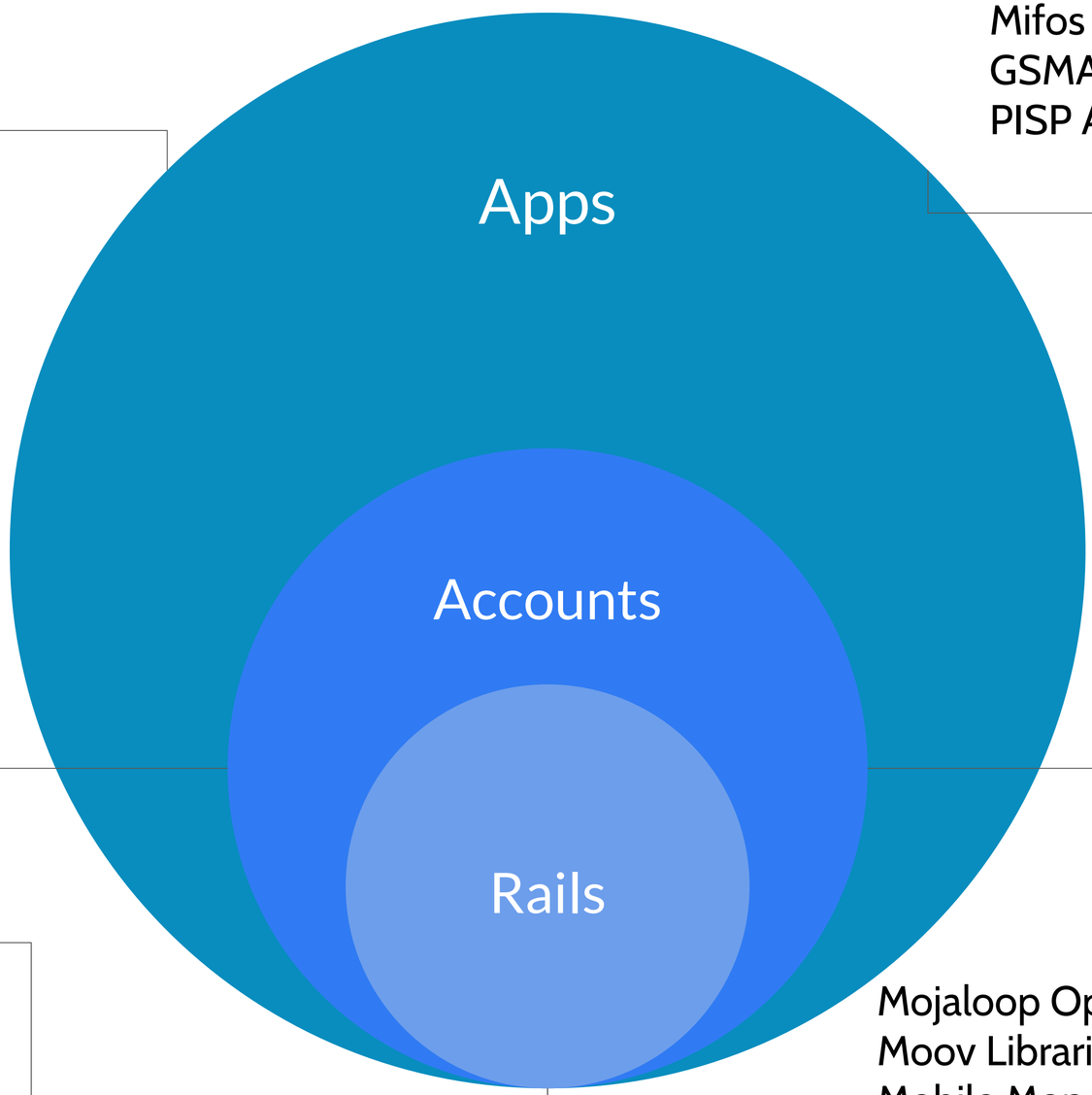
DPG

Long Tail of Adoption and Innovation - Mifos on Mojaloop Rails

Fintechs/PISPs

DFSPs

Switch Operator



Mifos Open Banking API
GSMA MM API
PISP API

Mifos/Fineract
OS Core Banking API

Mojaloop Open API
Moov Libraries
Mobile Money APIs

Origins & Evolution of PH-EE

Origins of Payment Hub EE

Led by Mifos Initiative with funding from:

2019

Prototype

dial Digital Impact Alliance

2020

Production-Ready Architecture

Bill & Melinda Gates foundation

dpc consulting

2021

Deployment & Scalability

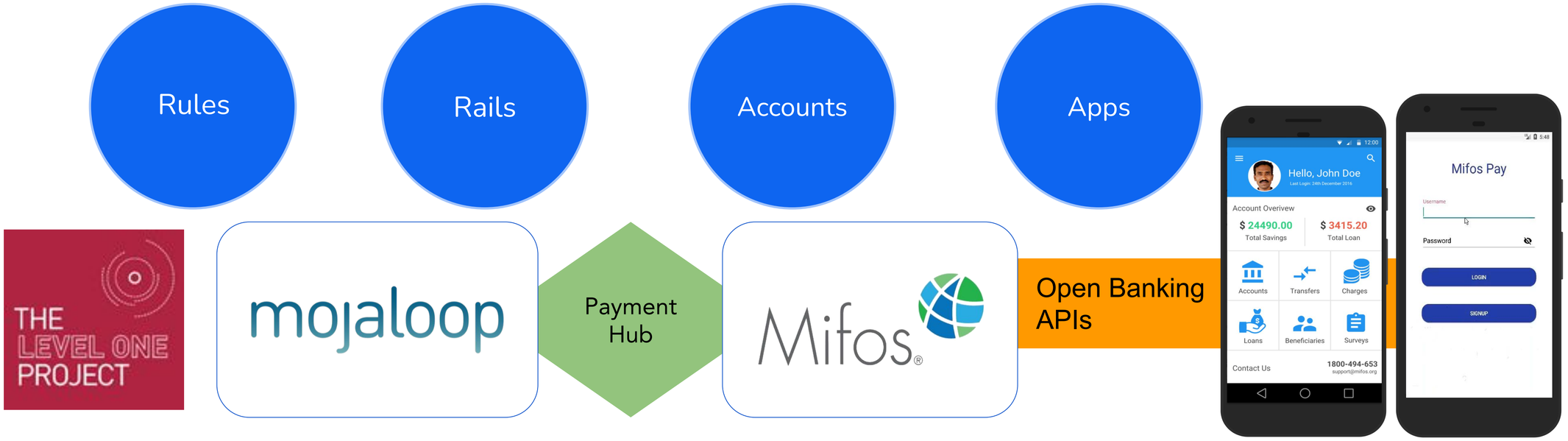
FINTECHEANDO

Fynarfin

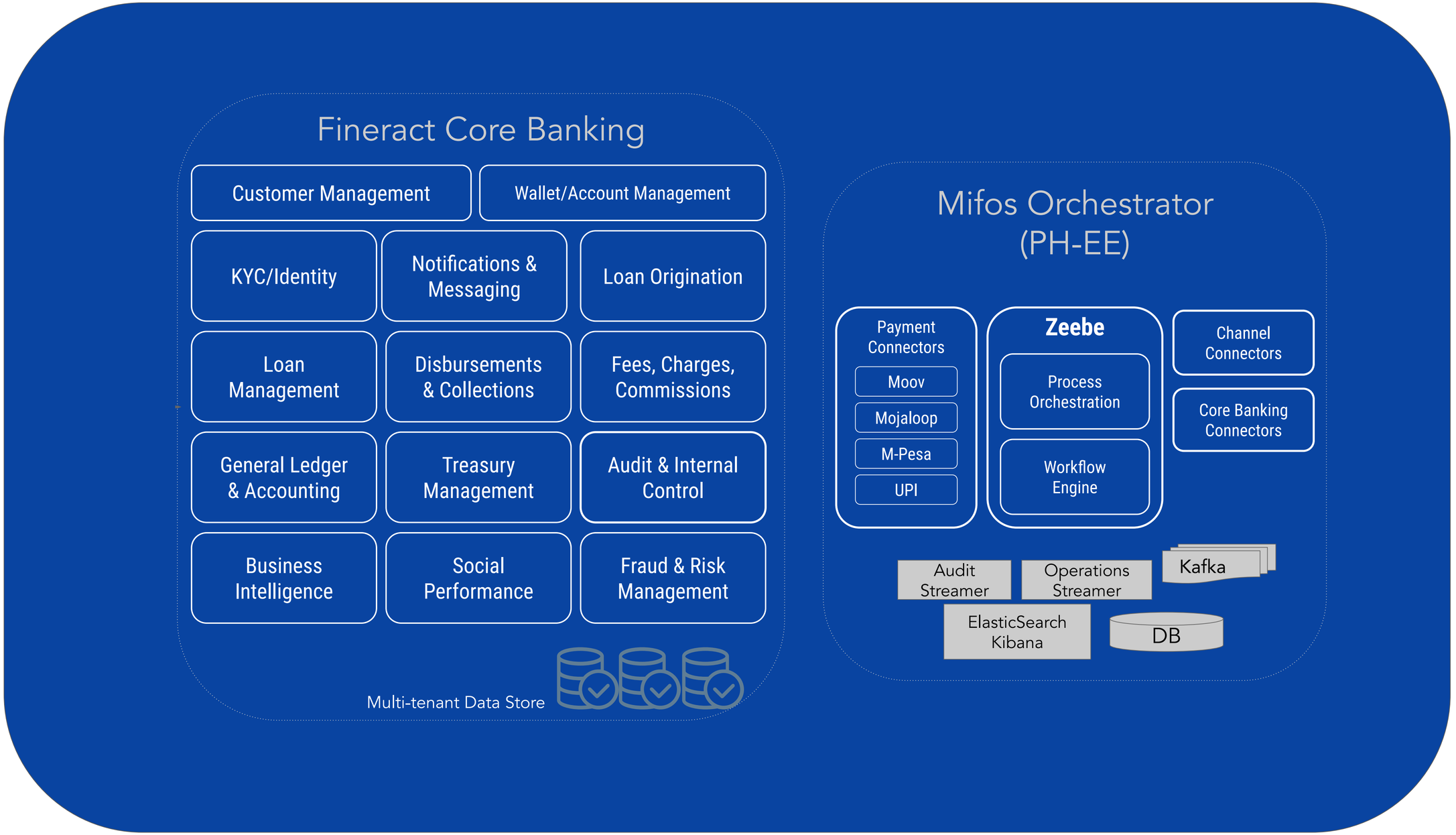
dpc consulting

End to End Open Source Architecture for DFS

- OS Payment Switch - Mojaloop | OS Bridge - Payment Hub
- OS Account Management - Mifos/Fineract | OS Reference Mobile Apps - Mobile Wallet/Banking



Payment Hub EE & Mifos/Fineract

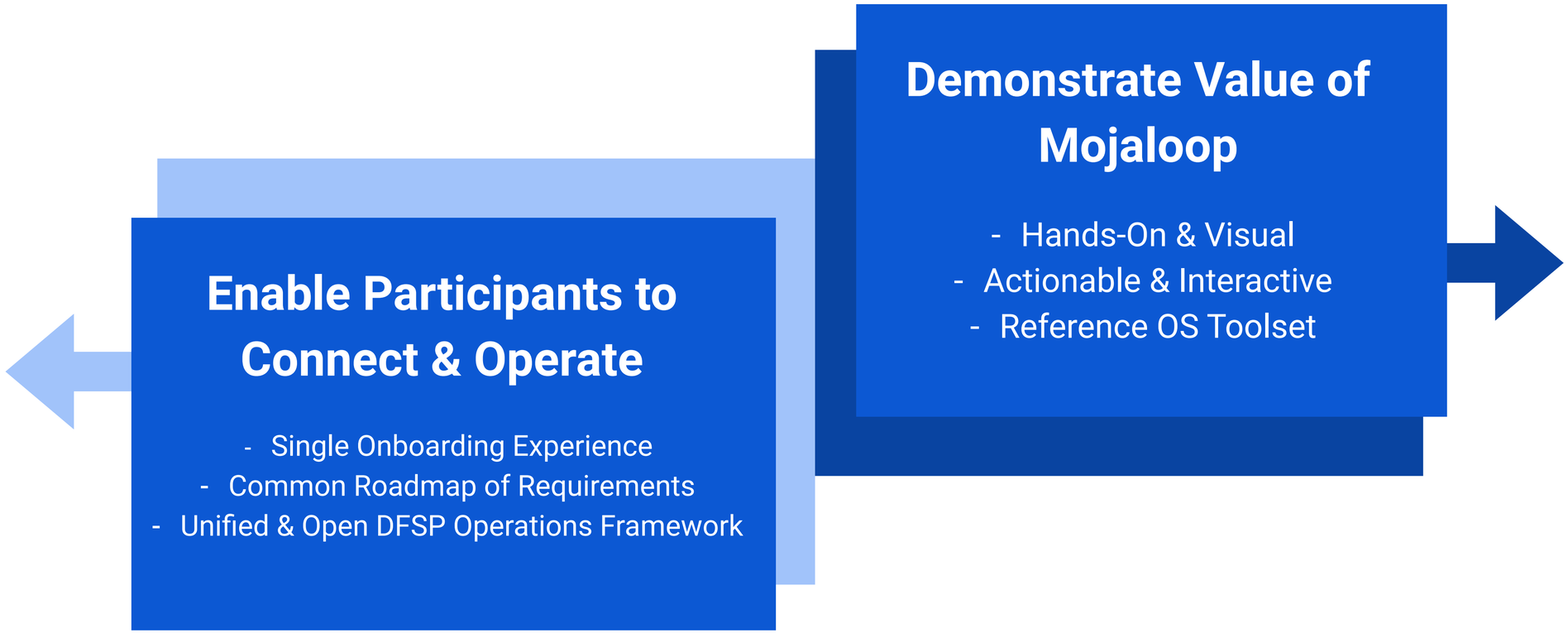


Payment Hub EE Overview

What is it and why it matters.

Community Need

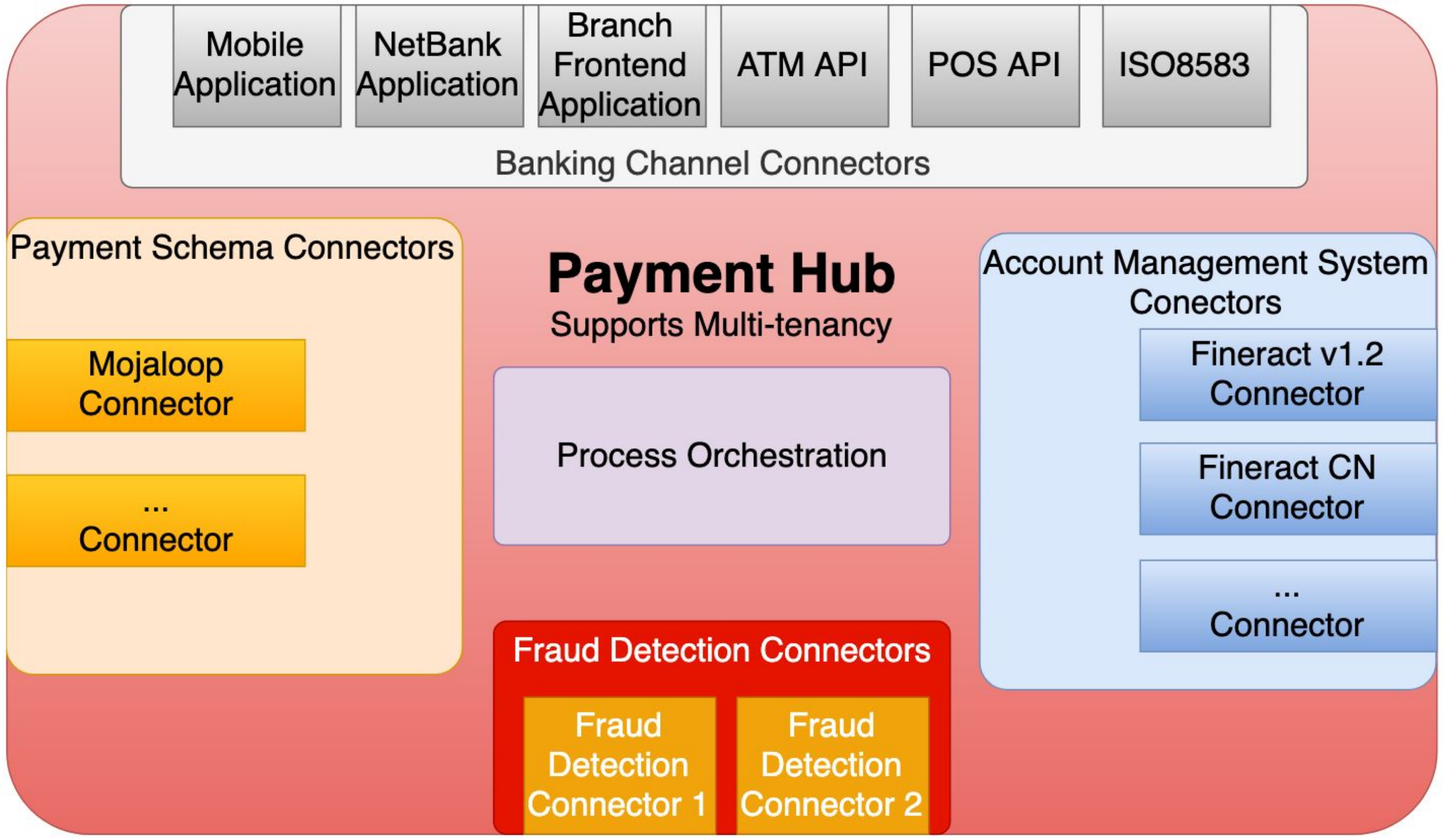
Accelerate Adoption of Mojaloop



mojaloop foundation

Payment Hub as an Open Source Asset for the Community

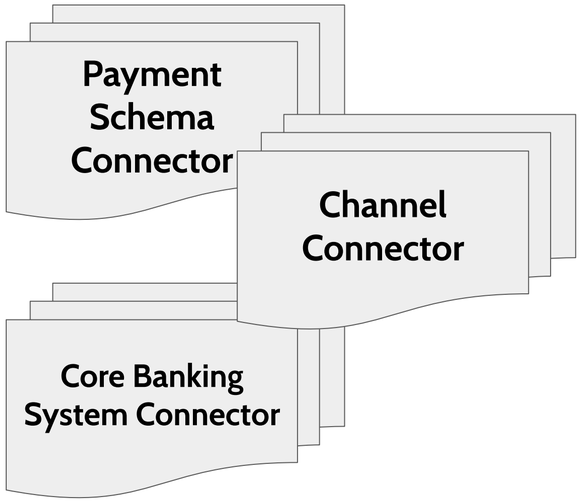
- The role of a payment hub to connect:
  - Financial Institution channels (Mobile, Internet, Branch, Callcenter, ATM, POS, API Gateways)
  - Account Management Systems (AMS / Core banking platform), optionally fraud monitoring tools
  - Payment Schemes, such as Mojaloop
- Need
  - Consistent Way to Connect to Mojaloop
  - Effective Operational Participation
- Additional Capabilities
  - DFSP-level fraud monitoring
  - Bulk Transfer Campaign Management
  - Operational Monitoring
  - Manages the identifier – account relation
  - Trigger notifications
- Built on proven open-source technology:
  - Java, SpringBoot, Kafka, Elasticsearch
  - Apache Camel, Camunda Zeebe
  - Kubernetes



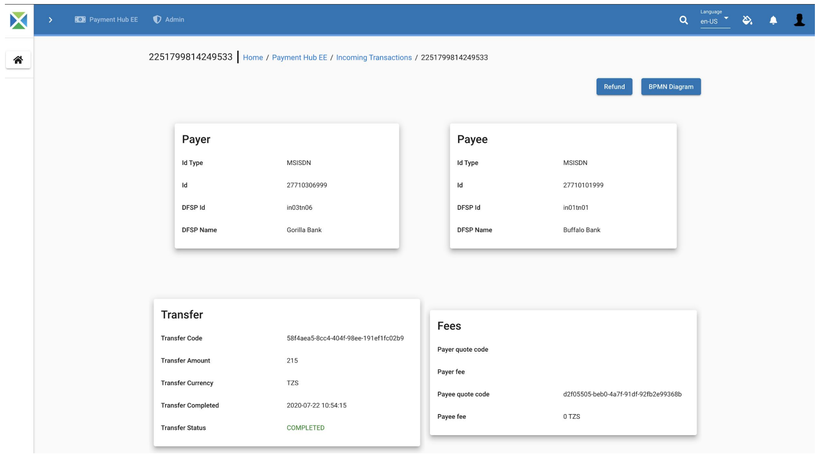
mojaloop foundation

What does Payment Hub EE do?

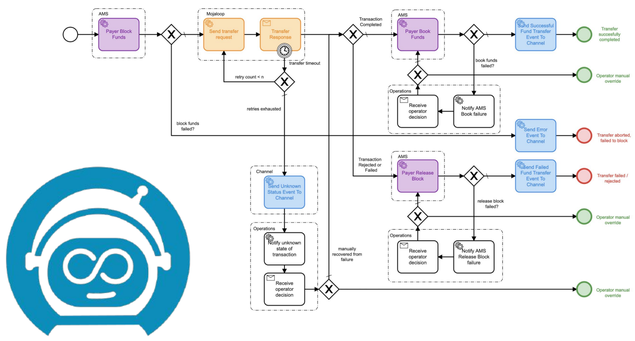
Integration



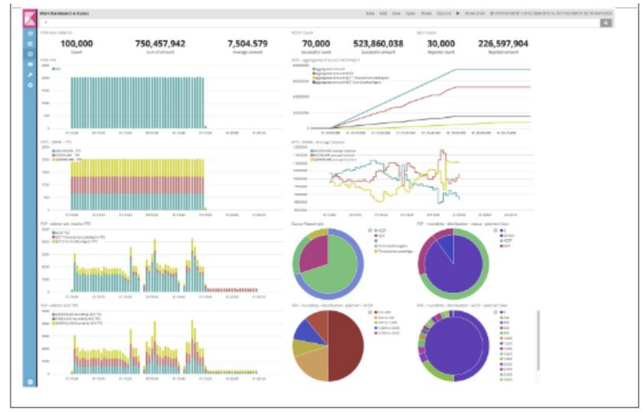
Operational Controls



Orchestration

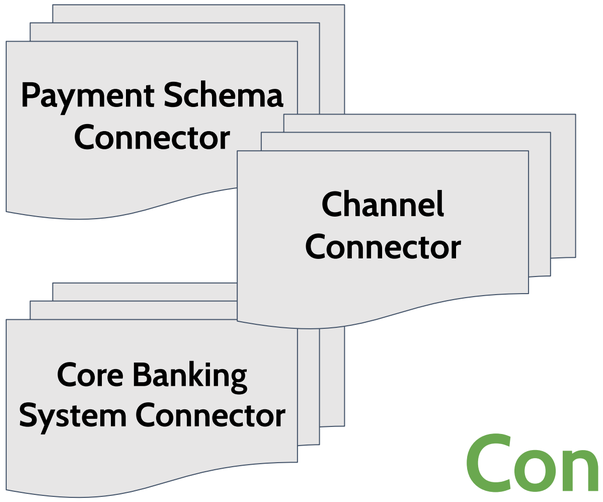


Analytics

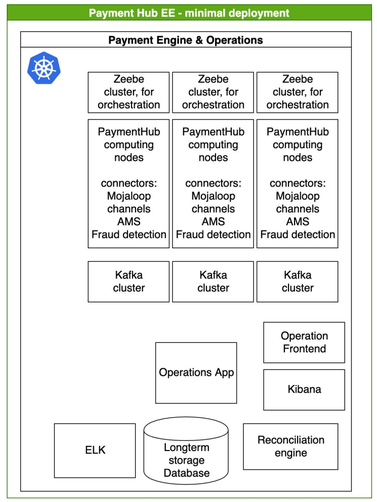


Benefits of Payment Hub EE

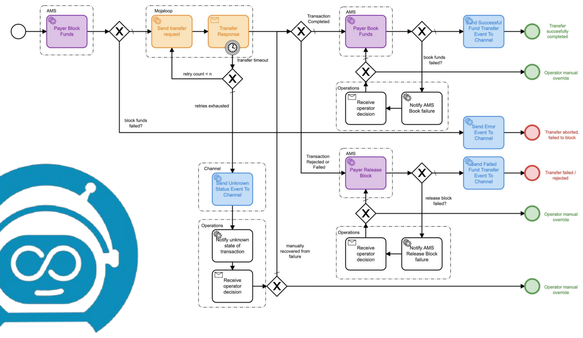
Extensible



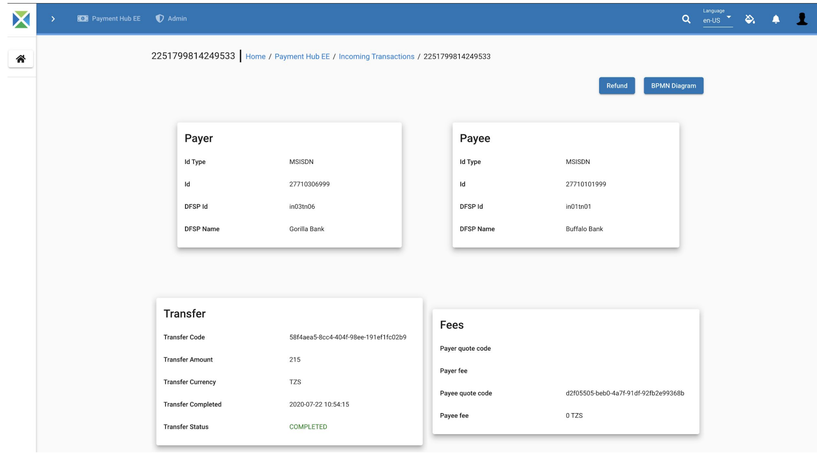
Scalable



Configurable



Practical



mojaloop foundation

Benefits of Payment Hub EE

- Ease of integration with external payment systems
  - Seamless & Easy to Use - Abstracts out complexity of API integration
  - Operational Control Center - Enables DFSP to monitor, analyze & respond to payment transactions in real-time
- Scalable & Enterprise-Grade
  - Flexible deployment options - fully containerized, PH-EE can be deployed on-premise or in the cloud
  - Different deployment scenarios: 3 different configurations based on the level of availability, fault tolerance and transaction volume
    - barebone | medium | fully scaled
  - Multi-tenant: Shared service providers offering services to multiple smaller DFSPs to spread out operational costs
- Flexible & Extensible
  - Core Banking System Agnostic
    - Connect to any core banking or account management system simply by building another connector.
  - Flexible Payment Bridge
    - Connect to any other payment system simply by building additional connectors such as SEPA, SWIFT, ISO 20022, or any mobile money API.
  - Extensible Architecture
    - Simply build additional connectors for payments, core banking systems, or channel applications.
    - Orchestrate these different microservices by creating new BPMN diagrams and Zeebe workers.
- Payment and Process Orchestration
  - Zeebe: Powered by Camunda's modern workflow engine for microservices orchestration built by Camunda.

Accelerate DFSP adoption

Turnkey solution for digitization & digital transformation of MFIs & SACCOs

Challenge	Solution
MFIs lack processes & systems to participate in switch	Cloud solution for digitizing and automating core banking operations
MFIs lack systems with 24x7 capabilities	Payment Hub EE can provide stand-in processing
MFIs have wide variety of core banking systems and channels	Extensible by simply building additional core banking system or channel connectors.
MFIs learn in visual manner	Hands-on lab environment with actionable tools
Difficulty connecting dozens or hundreds of MFIs	Multi-tenancy of Mifos and Payment Hub EE enabling economies of scale for shared service providers
MFIs need support and training in getting regulated & connected	Mifos network of local on-the-ground integrators & support partners with deployment & domain expertise.

Tier 1 DFSPs

Robust Performance
Deployment Flexibility
Ease of Connecting
Operational Controls
Seamless Channel Integration

- Core Banking System Agnostic
- Highly configurable workflow engine
- Extensible via connectors
- Ready to customize and extend
- Vendor-agnostic
- Operational Control Center

Fintechs

Ease of Connection
Rapid Innovation
Sandbox
API Standardization
Use Case Prototyping

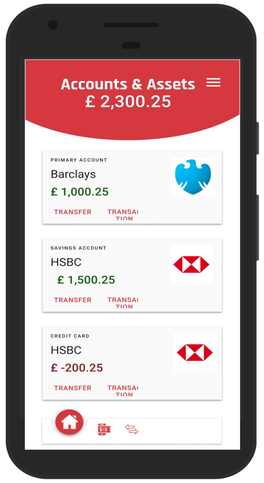
- OS Reference Apps
- Interactive Lab Environment
- End to End OS Stack for DFS
- Open Banking API Layer
- GSMA Mobile Money API support

Regulators

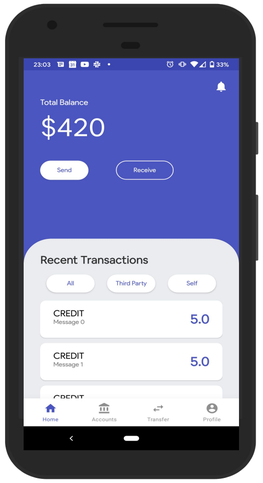
Transparency & Visibility
More input into Fraud Detection
Velocity of Cash

- DFSP-level fraud monitoring
- Visibility into on-us transactions
- Generate regulatory reports
- Consistency of data
- User Access for regulators

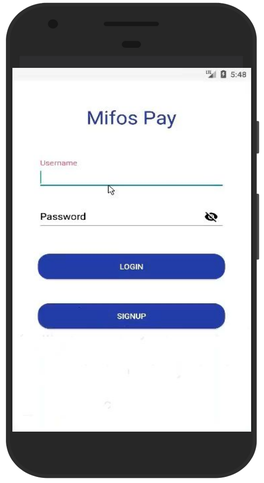
Interactive & Immersive Lab Environment



Open Banking App



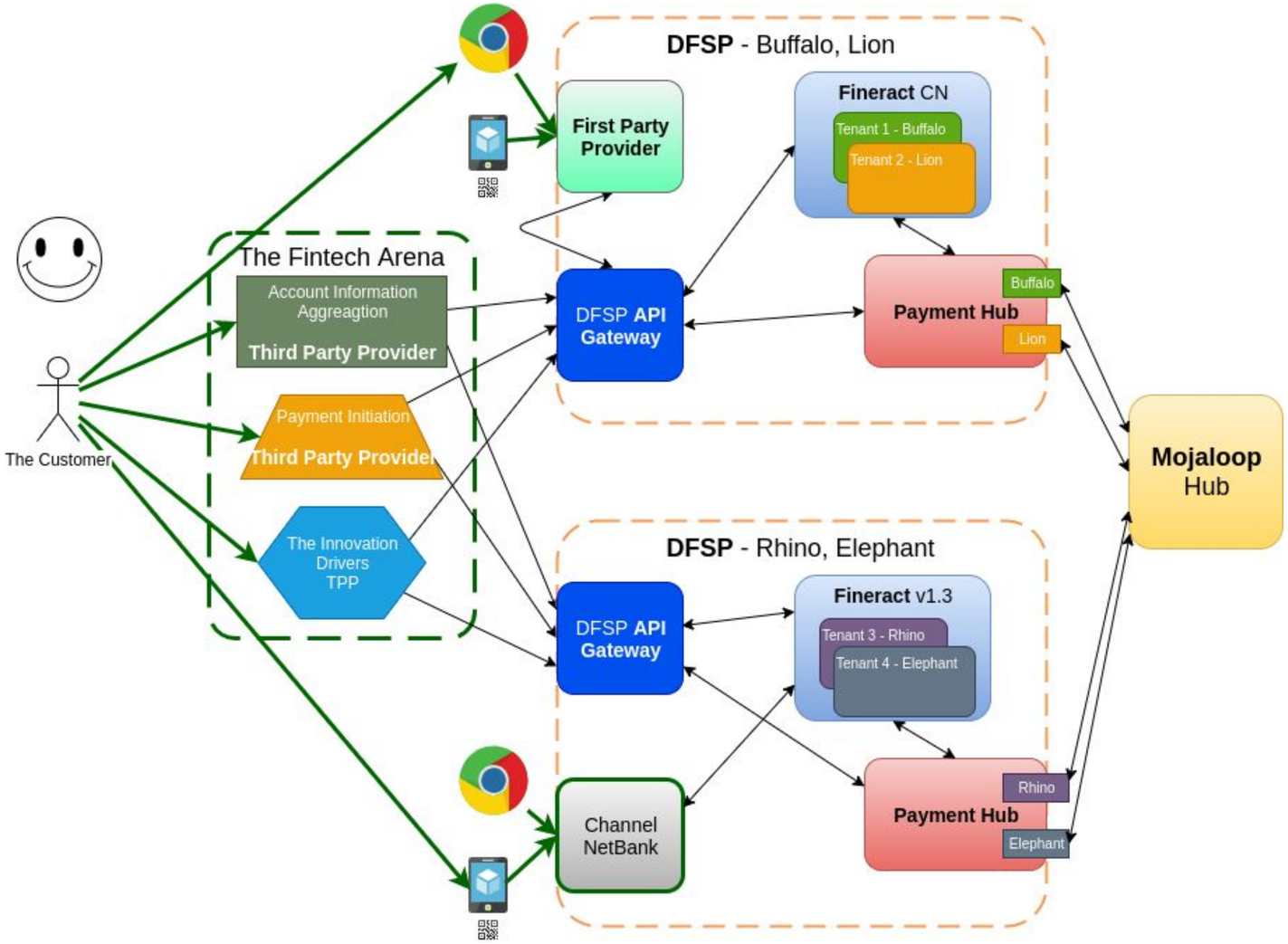
Mobile Banking



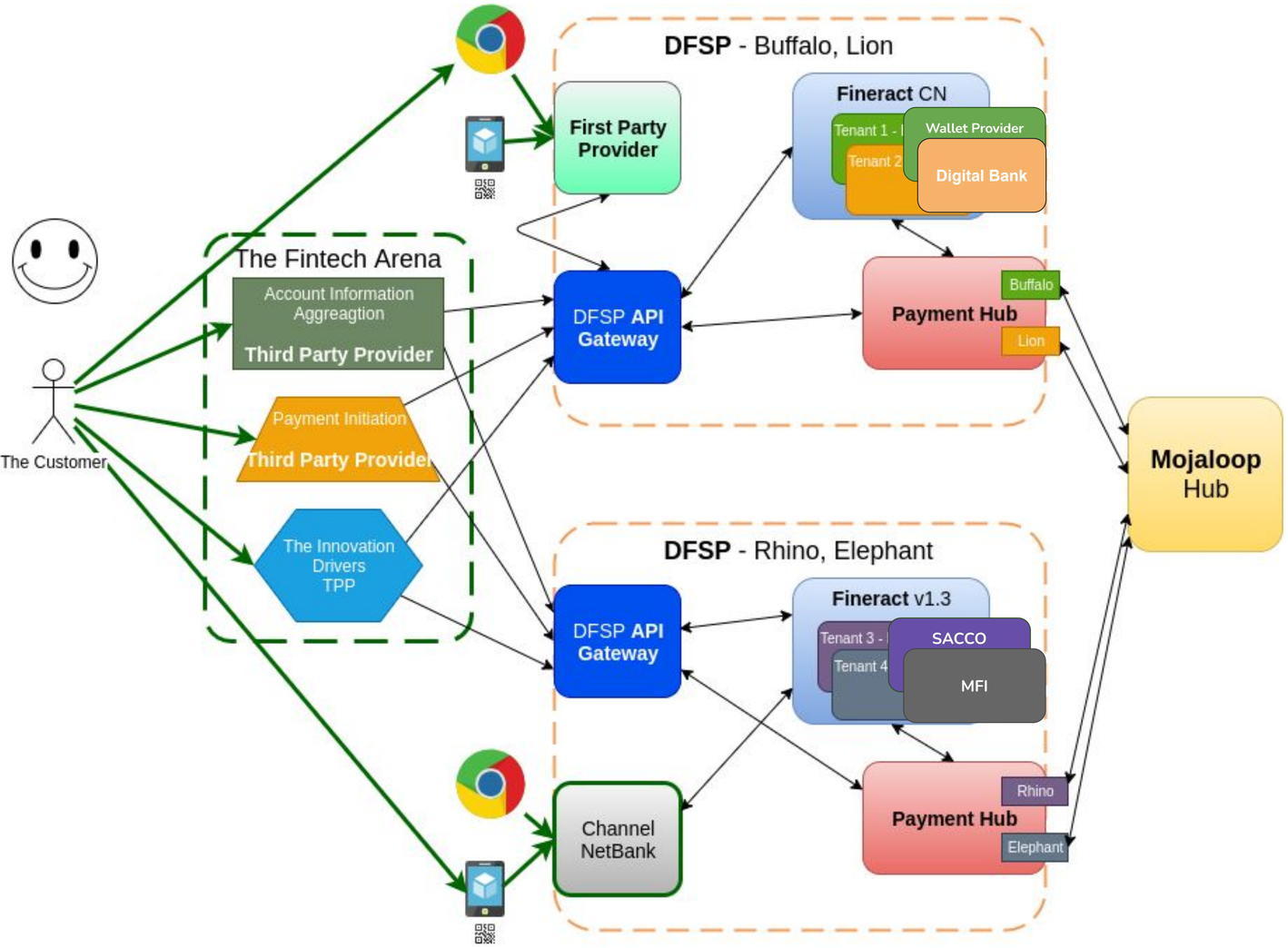
Mobile Wallet

Demonstrate Value of Real-Time Payments

- Hands-On & Visual
- Actionable & Interactive
- Reference OS Toolset



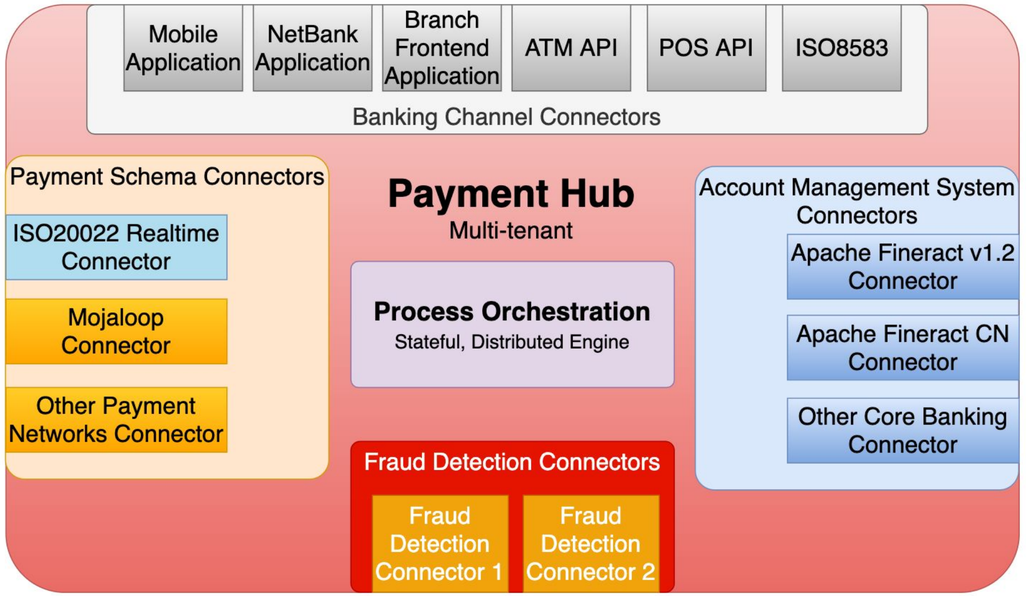
Sandbox Environment - Visual and Actionable

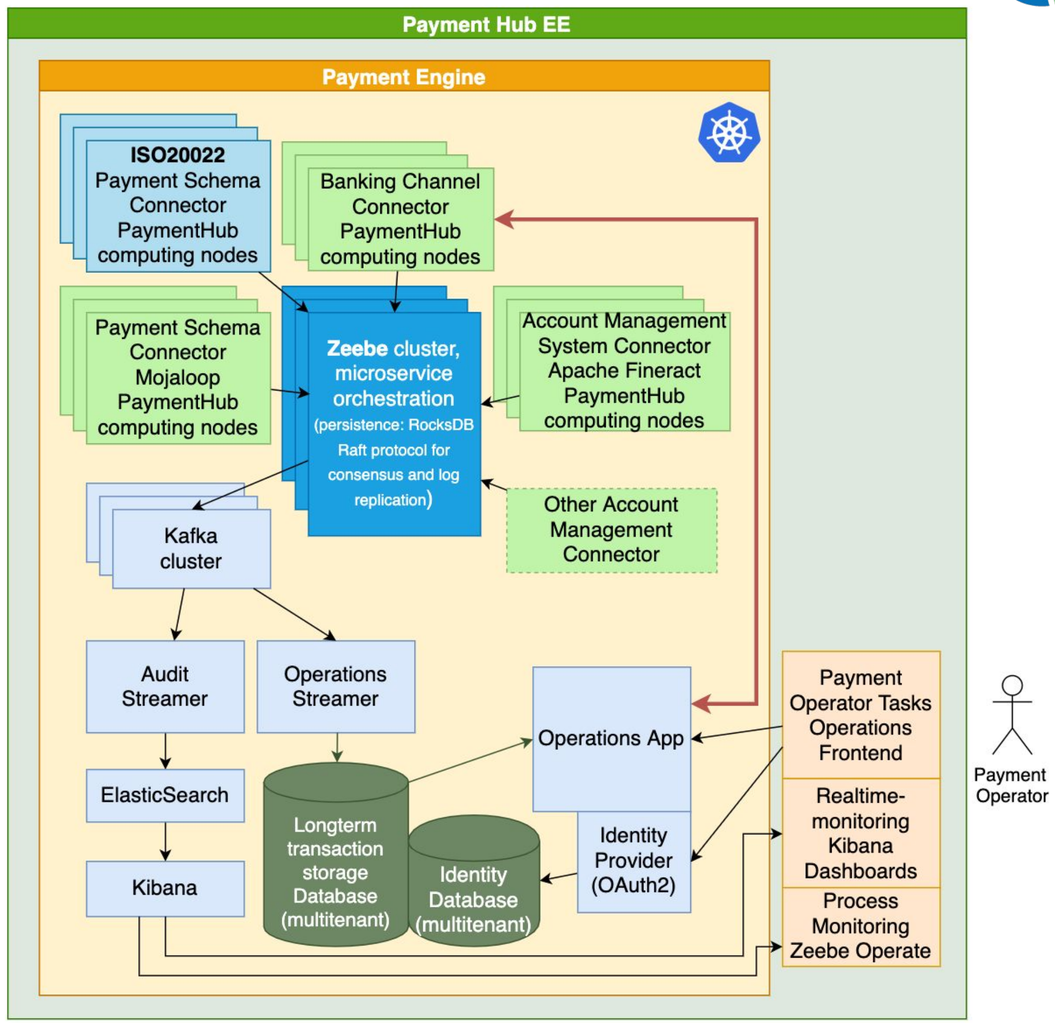


Audience	API Layer	OS Toolset
Fintechs	**- Open Banking API Layer** **- GSMA Mobile Money API (in progress)**	➢ Mobile Wallet App ➢ Mobile Banking App ➢ Online Banking App ➢ Open Banking Fintech App
DFSPs Banks, MFIs, SACCO, MMOS	**Mifos/Fineract API Layer** - Identity & KYC - Wallet/Account Management - Loan & Savings Management - Accounting & Ledger - Reporting	➢ API-Driven Open Source Core Banking Platform ➢ Angular Web UI for staff ➢ Android Mobile UI for staff
Hub Operators, Regulators	**Mojaloop API Layer** - Peer to Peer - Merchant Proximity Payment - Merchant Request to Pay - Bulk Payment/Transfer (upcoming)	➢ Payment Hub EE ○ Operations UI ○ BPMN Workflow Engine

Technical Benefits

Payment Hub EE Architecture





Built on proven open source technology:

- Java, SpringBoot, Kafka, Elasticsearch, Apache Camel, Camunda Zeebe, Kubernetes

Logical structure:

- Realtime engine, with microservices, and microservice orchestration for stateful process management
- Asynchronously separated audit log collection, and operations UI for payment operators manual intervention and automated reconciliation

https://github.com/openMF?q=ph-ee

Design Considerations

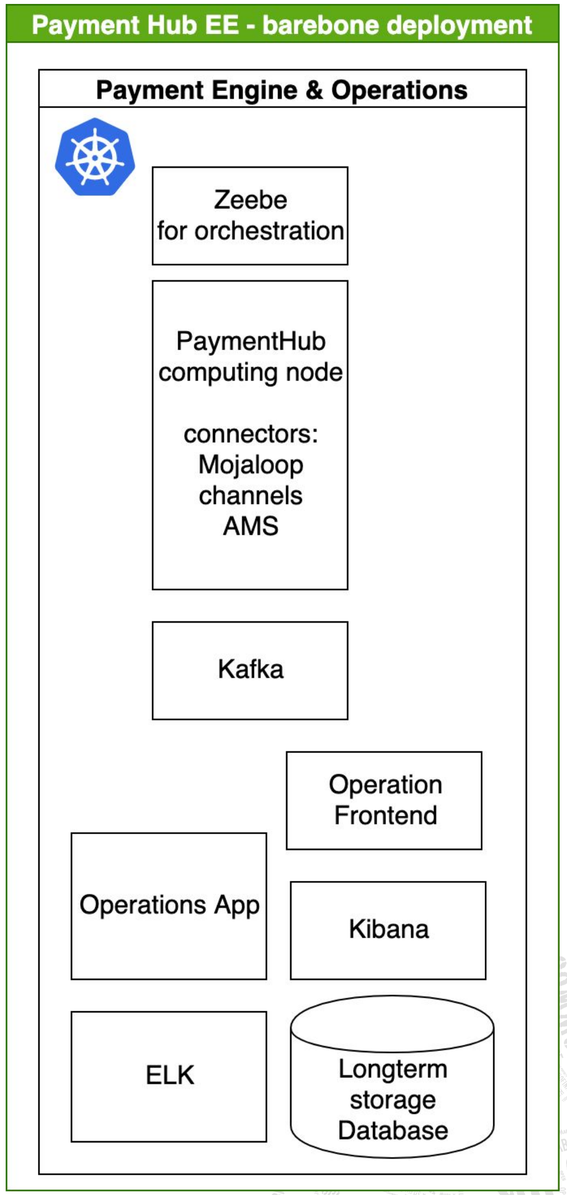
- **Connector components are stateless, easy and simple implementation**
- No additional components required in the realtime engines, no databases to maintain, all active process states are stored in the embedded RocksDB data store. Auditlogs, and other events are stored in Kafka clusters before storing them
- Scalability - using partitioning (sharding), the system scales horizontally to create thousands of workflow instances per second
- Fault tolerance enables, that system recovers from a failure without data loss, and using replication to define the number of copies for all instances
  In case of failures, “leader election” is taking place for the given partitions to continue operations uninterrupted (using Raft Consensus and Replication Protocol)

What is important?

- Payment flows surviving pod restarts (partial service disruption)
- Timers surviving node failures without interruption
- Correlation of incoming messages to existing workflow instances across the cluster nodes
- Capability to handle non correlating cases - important for cross payment hub engine flows
- Handle failover situation without interruption (3 nodes: 1 can fail, or 5 nodes: 2 can fail) - utilizing consensus in Zeebe, Elasticsearch, Kafka and Kubernetes to hand over “leadership” in case of failed nodes and replication of state

Deployment models

- On-premises and any of the cloud providers
- Shared service serving multiple DFSPs run by an aggregator (multiple SACCOs, credit unions on a multitenant setup) or dedicated setup
- Depending on the DFSP requirements it could be deployed as
- **barebone** - single instance of components, minimized resource usage, no loss of functionality
  Might not run on a feature phone, but we will get there.
- **minimal** - single realtime engine
- **fully scaled** - multiple realtime engines
- The difference is in availability, fault tolerance and the volume of transactions, which can be handled.



What Has Changed

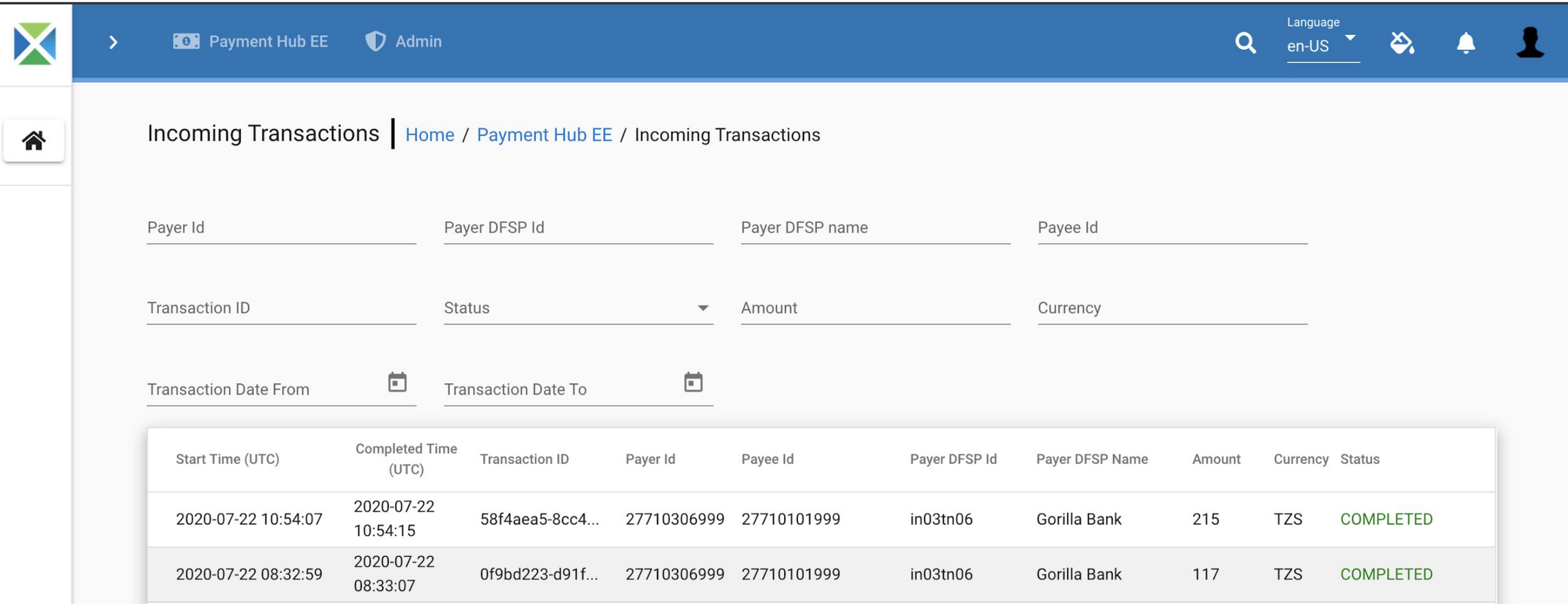
Enhancements to PH-EE since last update via upstream contribution

Where is Payment Hub EE being deployed?

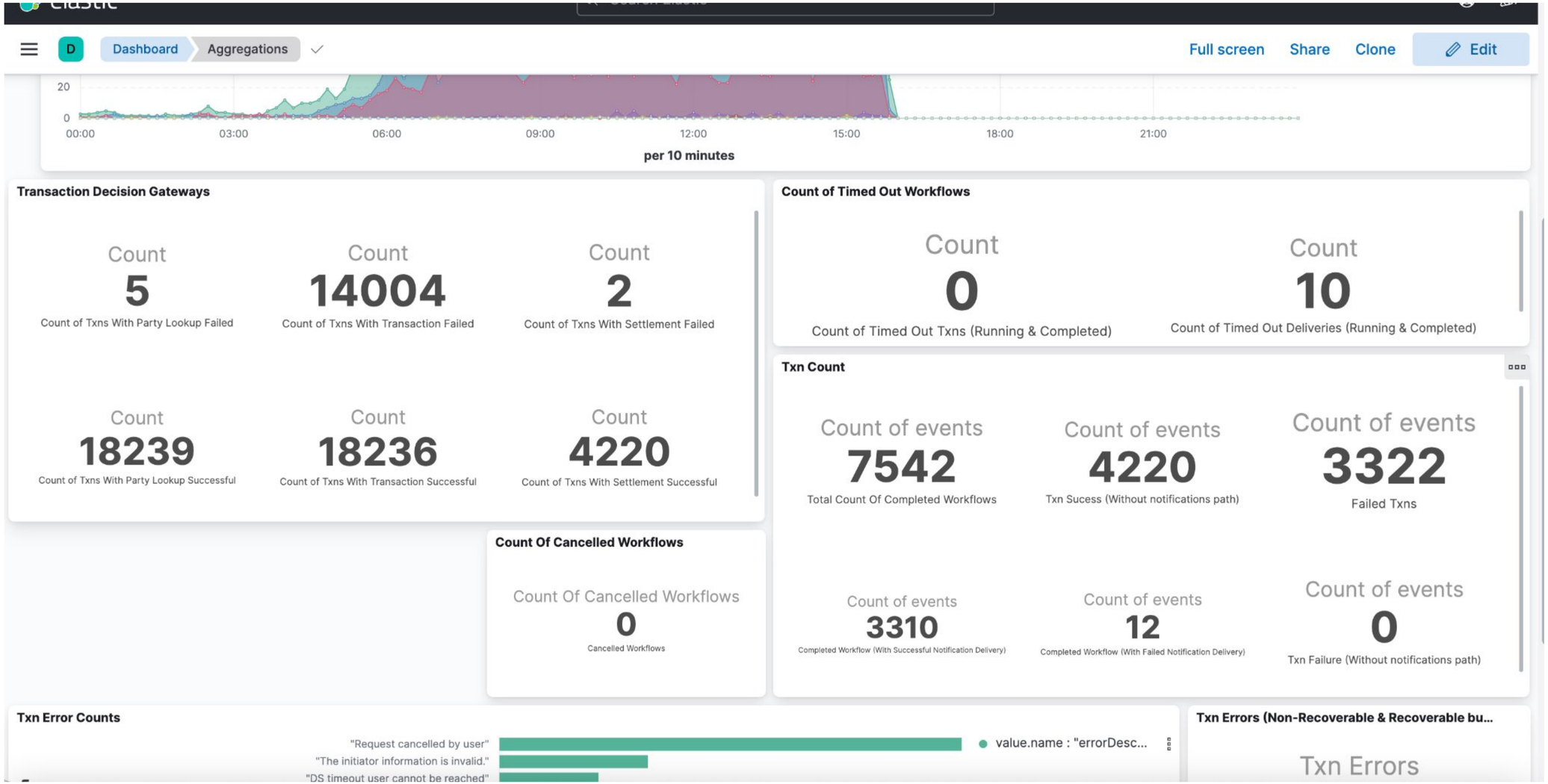
Upstream Contributions to PH-EE

- ❑ Demonstrating production-grade readiness
- ❑ Enhancements to Operations UI
- ❑ Message gateway support
- ❑ Monitoring Dashboards via Kibana
- ❑ Improved Deployability via Helm Charts
- ❑ Bulk Payment Pre-Processing
- ❑ Additional Payment Connectors

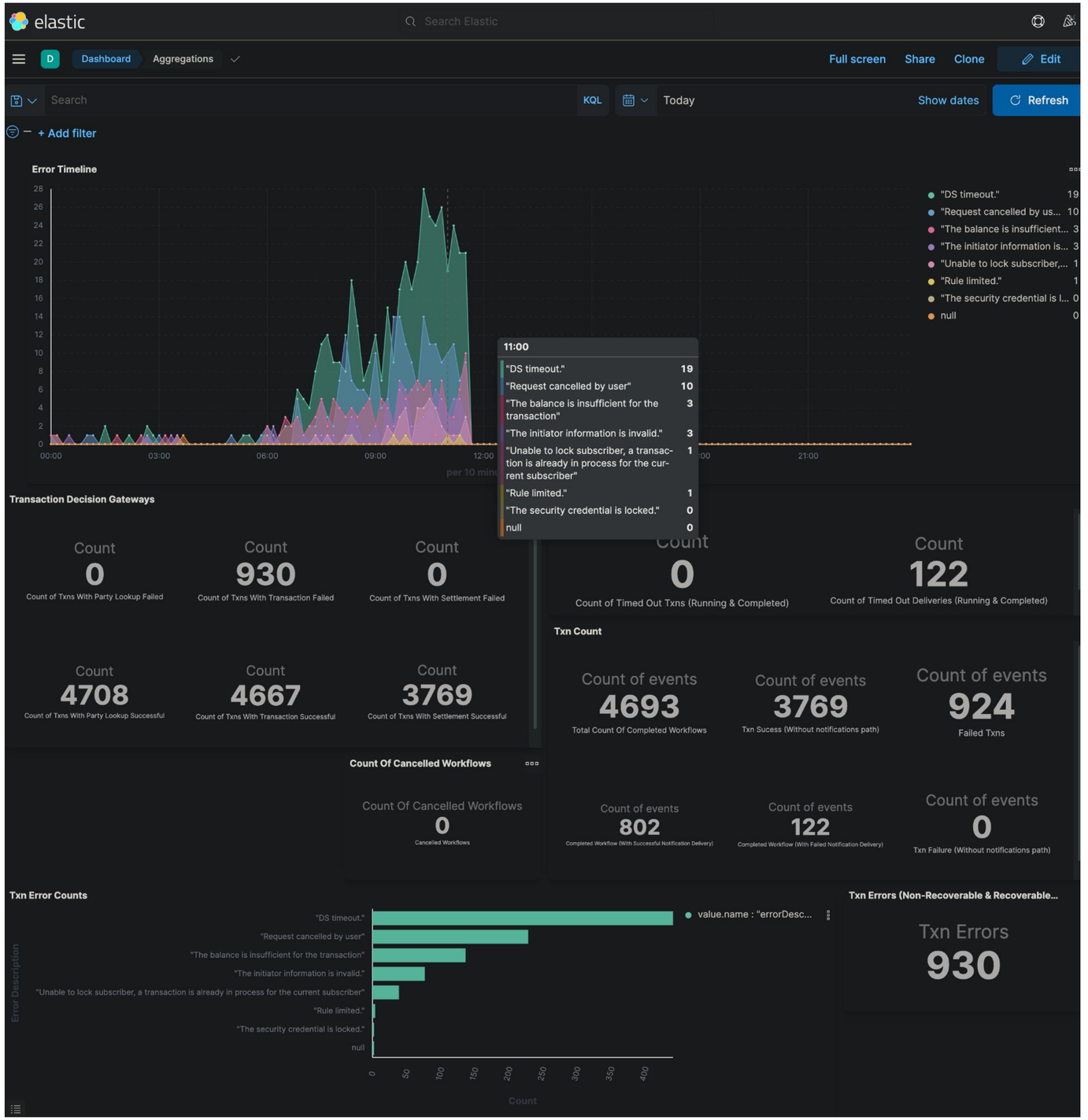
Operations UI Enhancements



Dashboards


Dashboard
Aggregations
Full screen
Share
Clone
Edit
20
0
00:00
03:00
06:00
09:00
12:00
15:00
18:00
21:00
per 10 minutes
Transaction Decision Gateways
Count
5
Count of Txns With Party Lookup Failed
Count
14004
Count of Txns With Transaction Failed
Count
2
Count of Txns With Settlement Failed
Count
18239
Count of Txns With Party Lookup Successful
Count
18236
Count of Txns With Transaction Successful
Count
4220
Count of Txns With Settlement Successful
Count Of Cancelled Workflows
Count Of Cancelled Workflows
0
Cancelled Workflows
Count of Timed Out Workflows
Count
0
Count of Timed Out Txns (Running & Completed)
Count
10
Count of Timed Out Deliveries (Running & Completed)
Txn Count
Count of events
7542
Total Count Of Completed Workflows
Count of events
4220
Txn Sucess (Without notifications path)
Count of events
3322
Failed Txns
Count of events
3310
Completed Workflow (With Successful Notification Delivery)
Count of events
12
Completed Workflow (With Failed Notification Delivery)
Count of events
0
Txn Failure (Without notifications path)
Txn Error Counts
"Request cancelled by user"
"The initiator information is invalid."
"DS timeout user cannot be reached"
value.name : "errorDesc...
Txn Errors (Non-Recoverable & Recoverable bu...
Txn Errors

Dashboards



mojaloop foundation

Journey of Use Cases

Existing Use Cases Supported

Integration & Orchestration Layer - Enabling Participation of DFSPs & Fintechs in IIPS

- P2P transaction
- Merchant proximity payment via QR Code
- Merchants request to pay flows
- Refund flows
- Third party initiation via Open Banking API

Payment Hub as Transformation Layer

- Cross-border remittances, PH-EE as bridge between ISO20022 and Mojaloop formats.

Additional Use Cases Supported

Integration & Orchestration Layer

- MFI Collection and Disbursement
- Merchant QR Code Payments

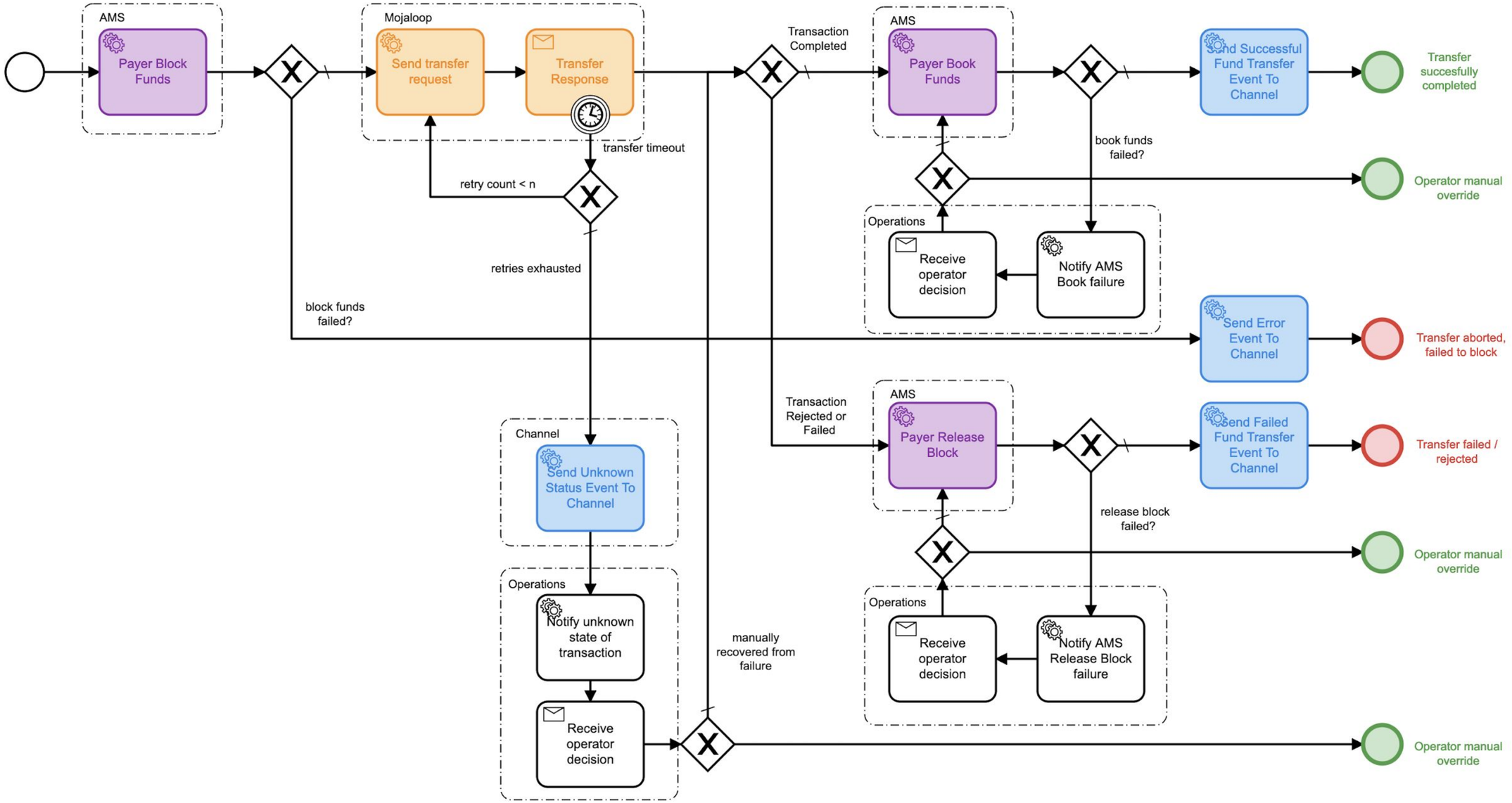
Payment Hub as Bulk Pre-Processor

- G2P/Bulk Payment Pre-Processing
- Identity-Driven Payments

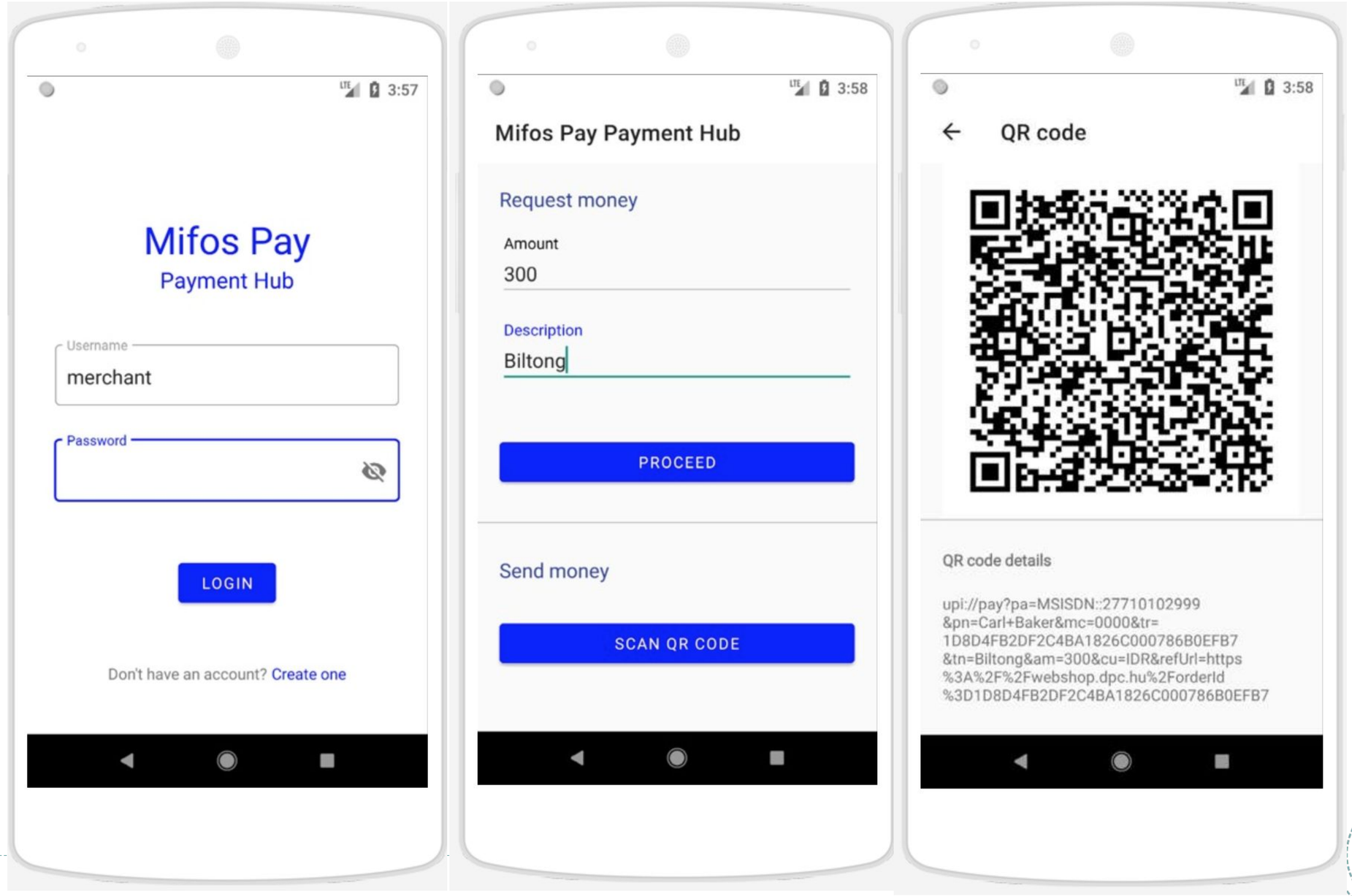
Payment Hub as Gateway Layer

- Stablecoins & CBDC

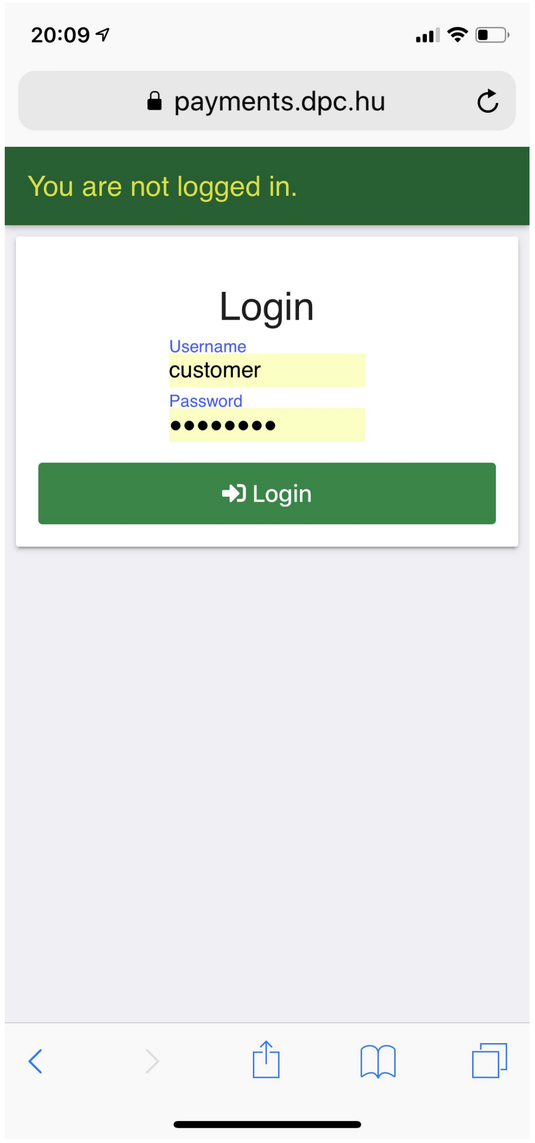
P2P Transfer (Payer Initiated)

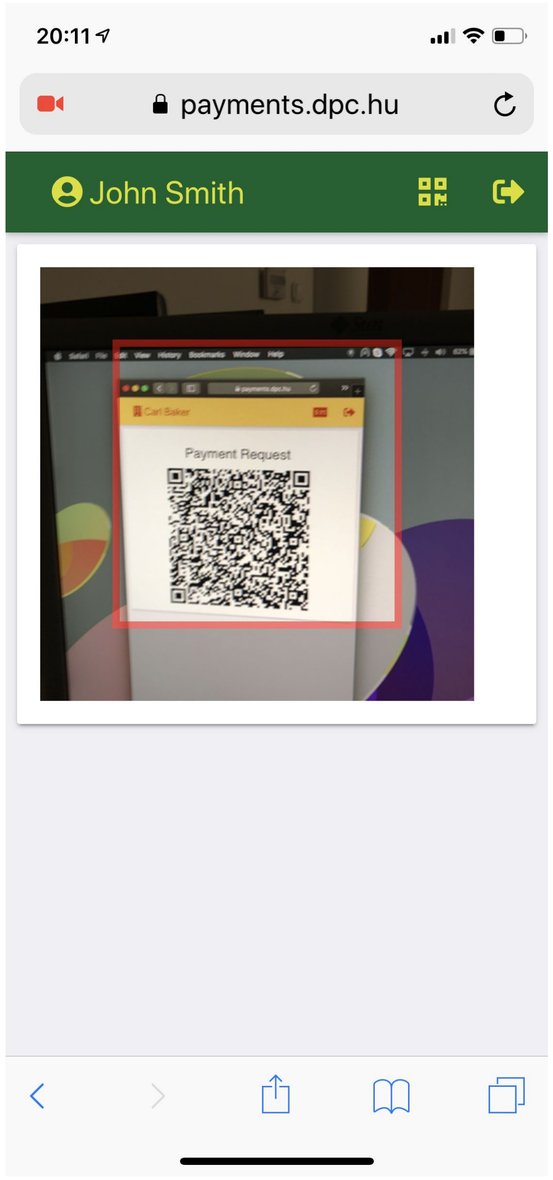


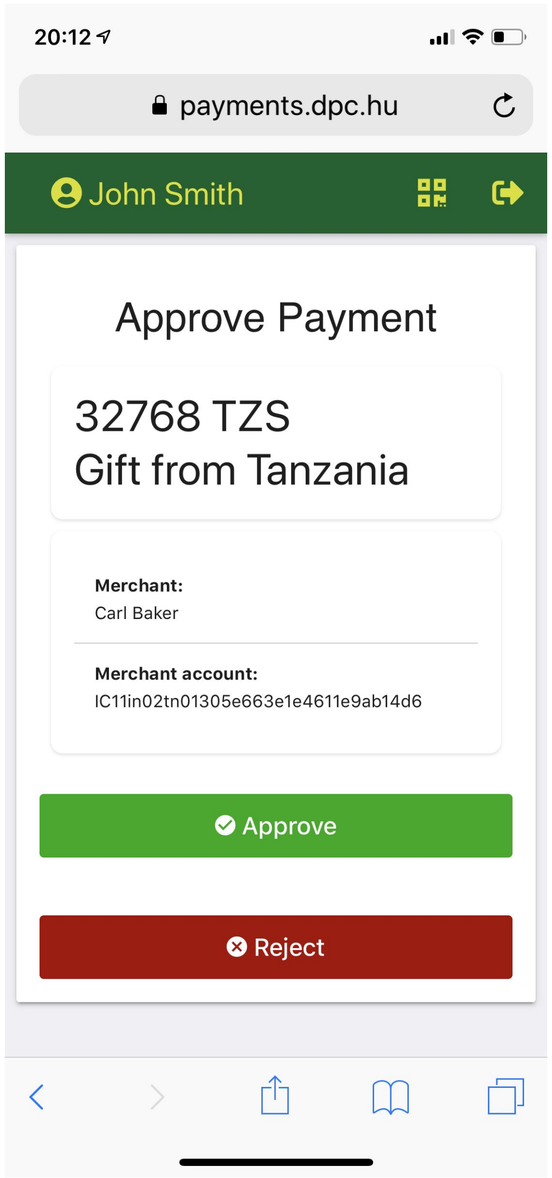
Merchant Prepares Transaction Using Android Mobile Wallet App

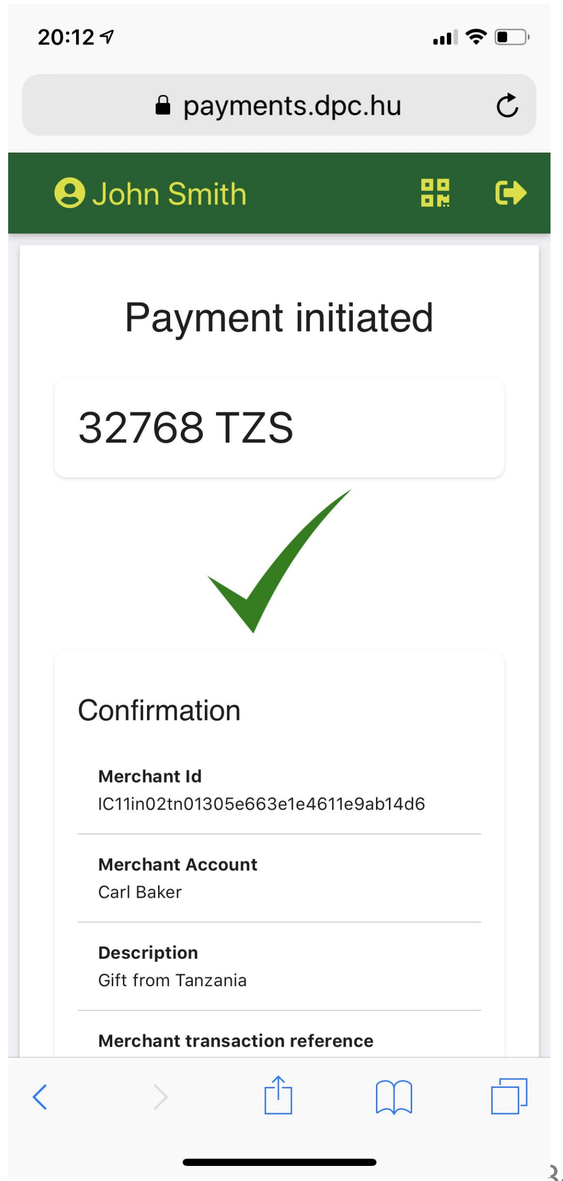


Customer Initiates Payment

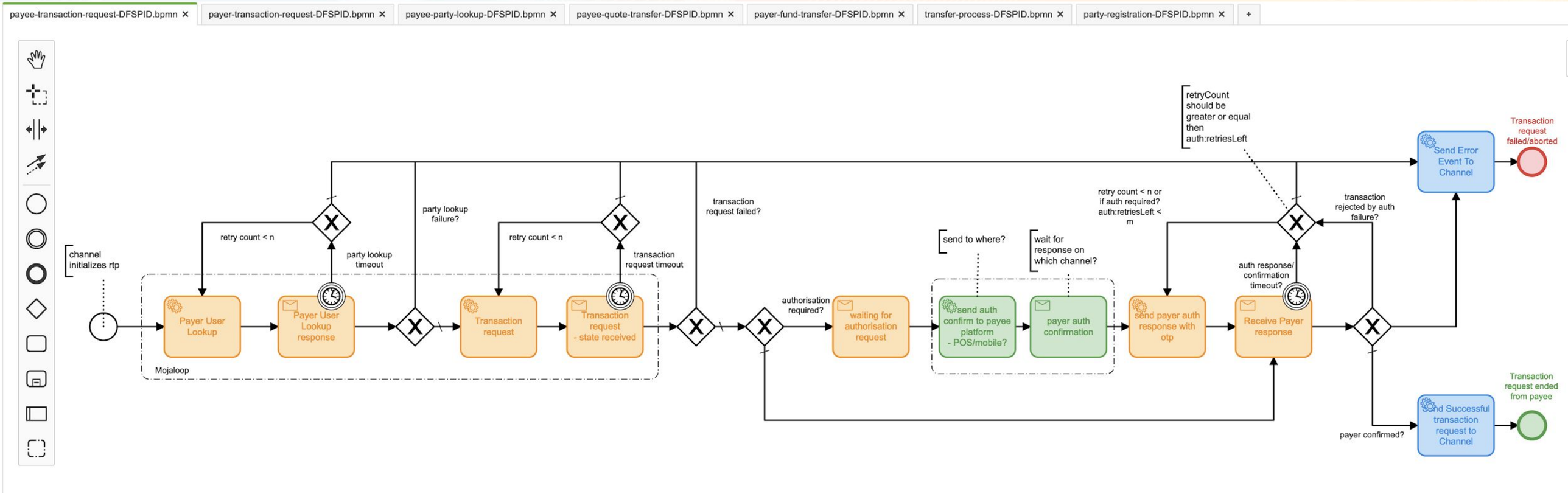




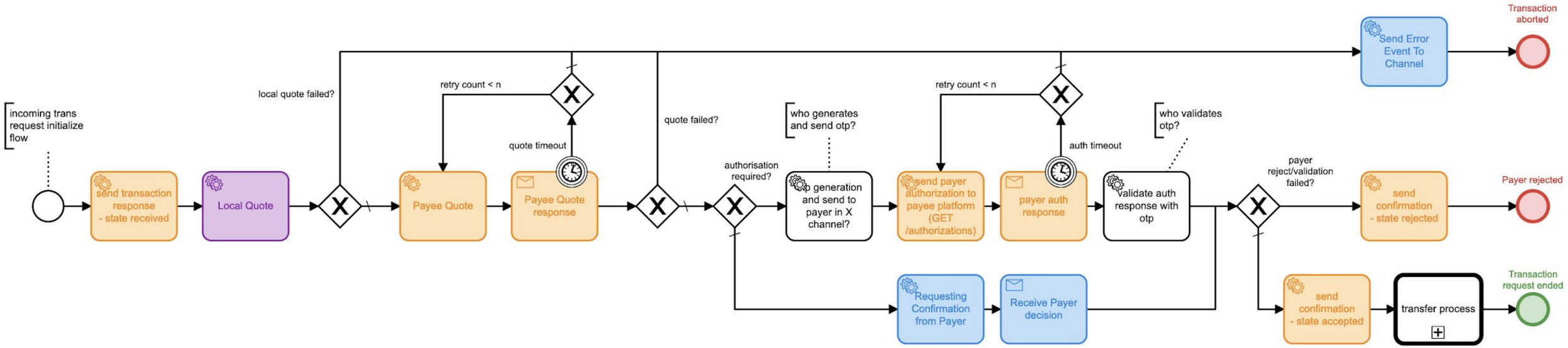




Payee Initiated flows - Request To Pay

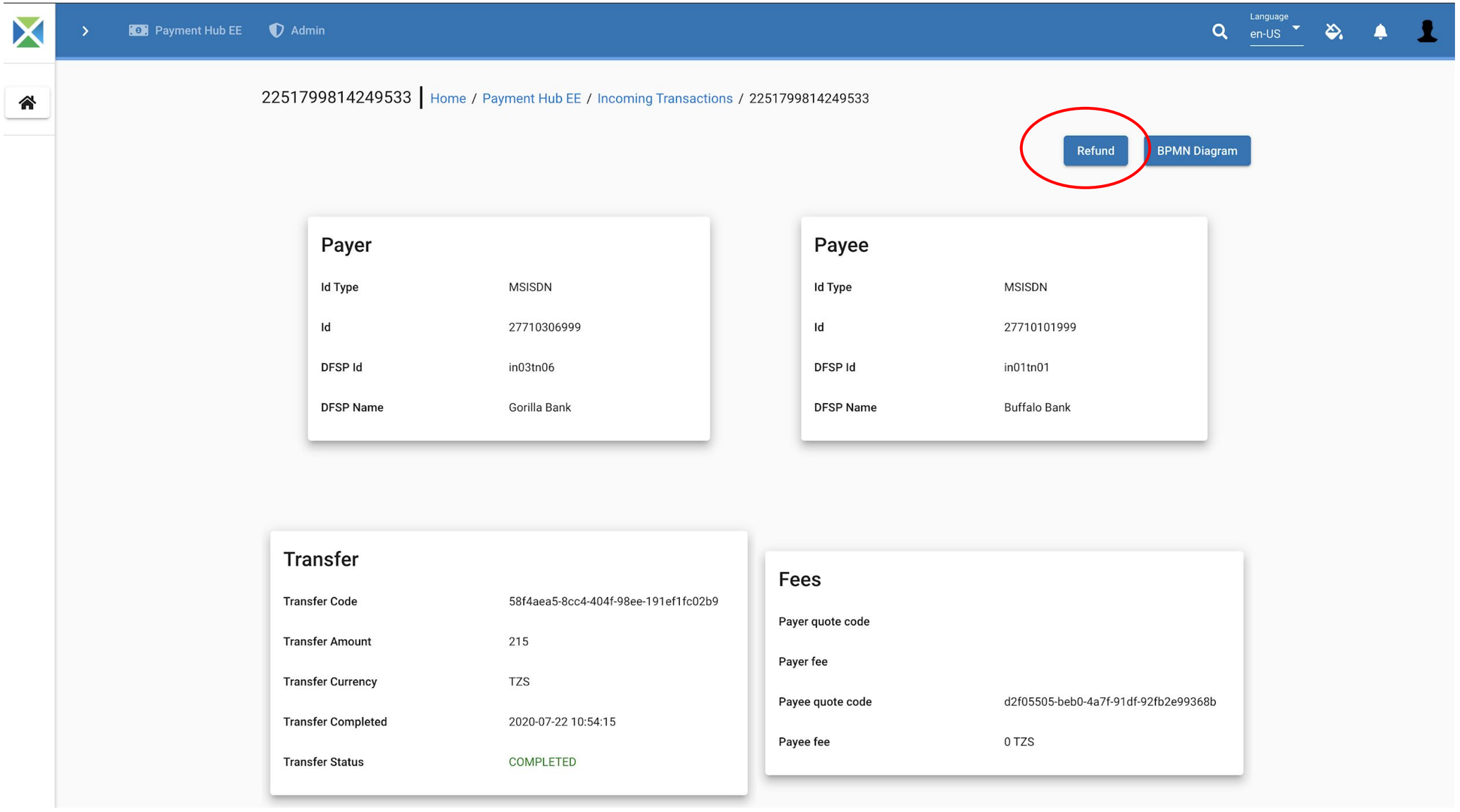


Sample PH-EE Flow - Accepting Request to Pay

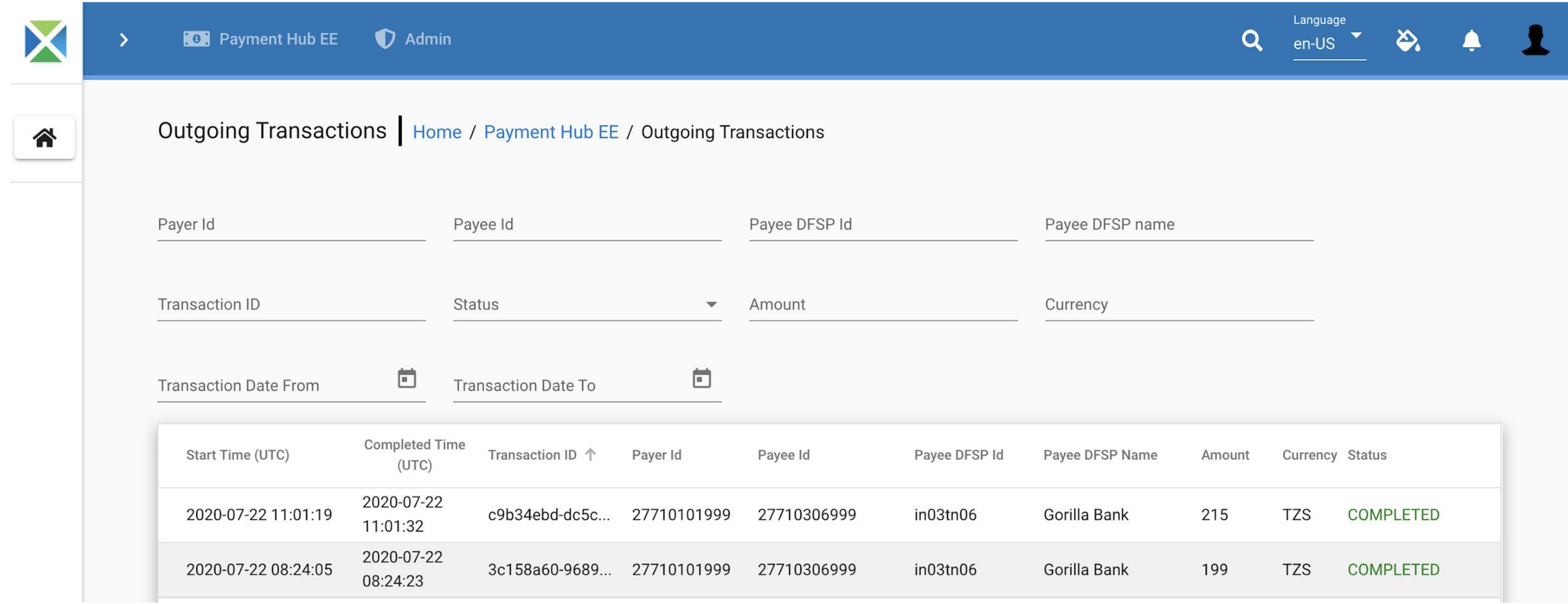


https://github.com/openMF?q=ph-ee

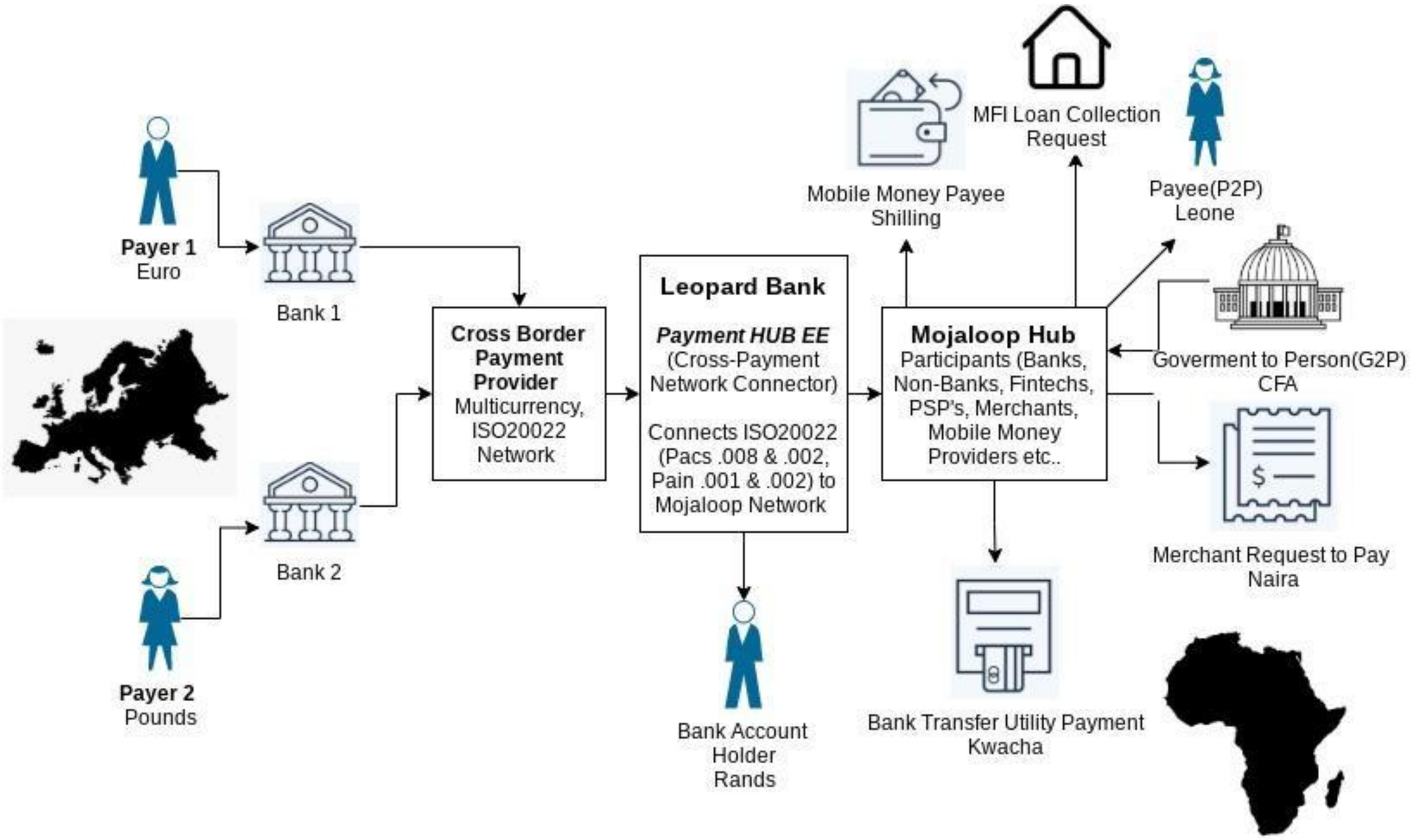
Incoming transfers with Refund capability



Outgoing Transfer - The Refund



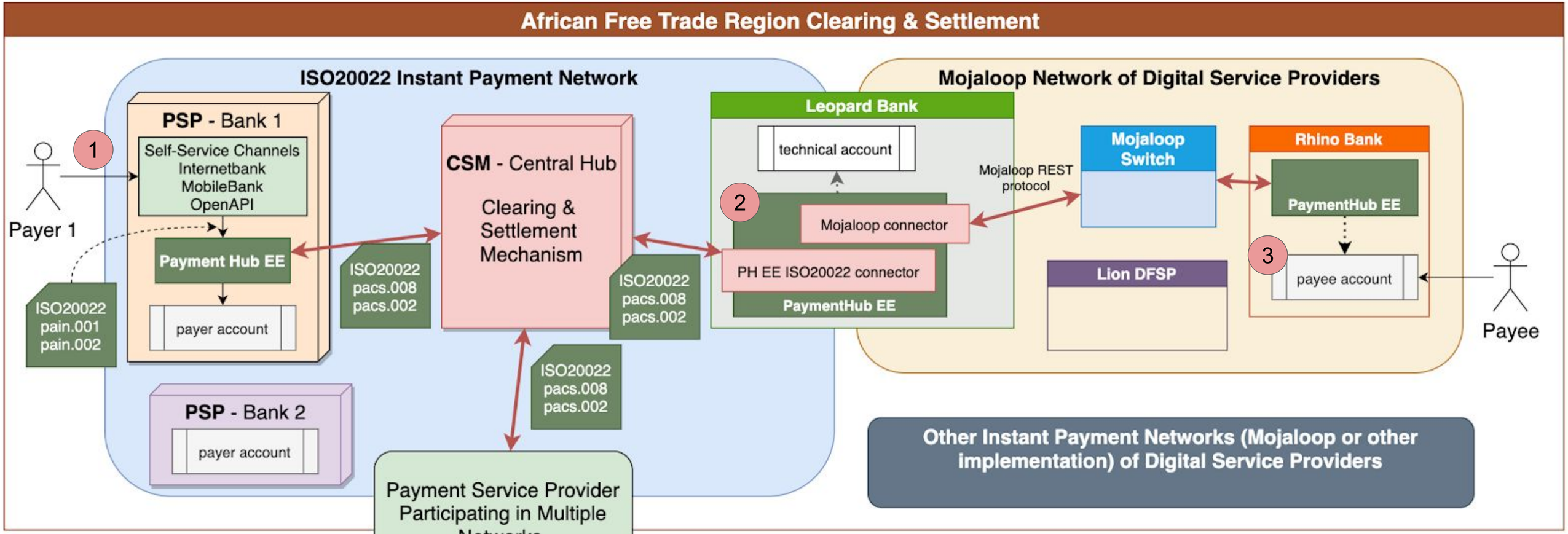
International Remittance & Last Mile Delivery Use Cases



Long-Term Vision:

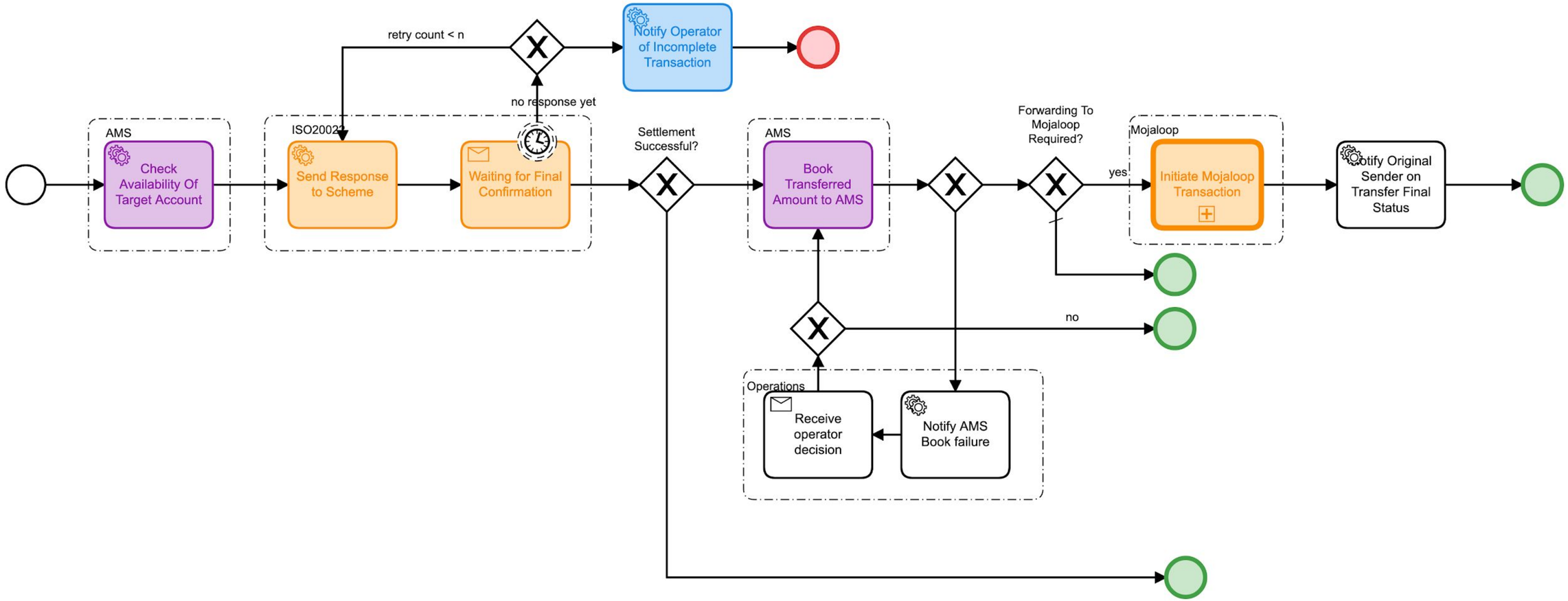
Leverage rich ISO 20022 data for more advanced use cases:

- **P2P: For-purpose remittances targeted for distinct expenditures**
- **B2C: Request to pay from merchants, billers, microfinance institutions.**
- **C2B: Dynamic QR codes and payments split across multiple vendors/billpayers**
- **B2B: Rich invoice data for SME payments**
- **G2P: Proper recourse and follow-through on payments to intended beneficiaries**



1) Payer 1 - using the channel of the bank, initiates a payment with pain.001 (mobile number of the Payee, provided using the Remittance Info of the messages) (the UI helps, to select the right institution and account at the intermediary bank (Leopard Bank) to reach the Payee)
2) Using pacs.008 and pacs.002 messages to complete an instant payment flow (based on SEPA Instant) funds arrive to a technical account at Leopard Bank in seconds, and transaction completes
3) Based on the Remittance Info, Leopard Bank identifies the Payee and its Financial Service Provider utilising the Mojaloop Network, and completes the second transaction
4) In case of failures, the necessary compensations will be executed by Leopard Bank

Crossnetwork BPMN flow

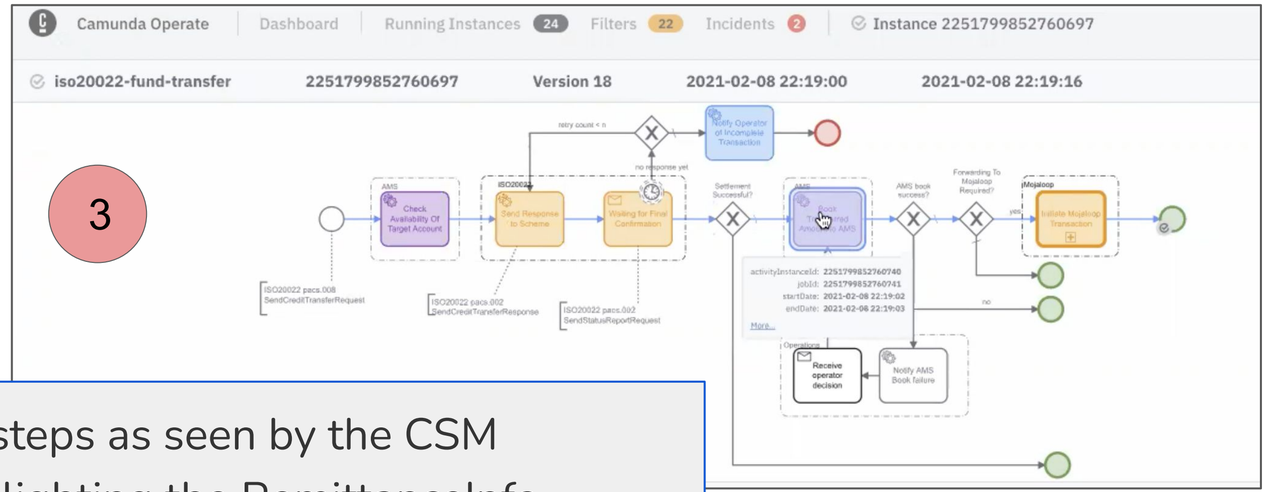


Demo Pt. 2 - Transformation of ISO message into Mojaloop format within Payment Hub EE

1

Description	ISO20022
CSM* -> Creditor sends Final Status Report (pacs.002)	pacs.002
CSM* -> Debtor sends Final Status Report (pacs.002)	pacs.002
Creditor -> CSM* received CreditTransferResponse (pacs.002)	pacs.002
CSM* -> Creditor sends CreditTransferRequest (pacs.008)	pacs.008
Debtor -> CSM* received CreditTransferRequest (pacs.008)	pacs.008

3



1. ISO instant transaction steps as seen by the CSM
2. pacs.008 message, highlighting the RemittanceInfo
3. BPMN flow in Zeebe microservice orchestration engine at the intermediary institution, showing the runtime execution of a transfer
4. Mojaloop Quote and Transfer messages, with the Interledger protocol digital signature

```
<?xml version="1.0" encoding="UTF-8" standalone=
<SendCreditTransferRequest xsi:schemaLocation="u
01.08 pain.001.001.08.xsd" xmlns="urn:eu:nets:re
iso:std:iso:20022:tech:xsd:camt.056.001.01" xmln
cs,004.001.02" xmlns:ns7="urn:iso:std:iso:20022:
n:iso:std:iso:20022:tech:xsd:pacs.002.001.03" xm
pacs.008.001.02" xmlns:ns3="urn:iso:std:iso:2002
ttp://www.w3.org/2001/XMLSchema-instance">
    <ns4:FIToFICstmrCdtTrf>
        <ns4:GrpHdr>
            <ns4:MsgId>02254AA26E8A48E9AF143B8ED
            <ns4:CreDtTm>2021-02-08T22:19:00.411
            <ns4:NbOfTxs>1</ns4:NbOfTxs>
            <ns4:TtlIntrBkSttlmAmt Ccy="HUF">512.00</ns4:TtlIntrBkSttlmAmt>
            <ns4:IntrBkSttlmDt>2021-02-08</ns4:IntrBkSttlmDt>
            <ns4:SttlmInf>
                <ns4:SttlmMtd>CLRG</ns4:SttlmM
            </ns4:SttlmInf>
            <ns4:PmtTpInf>
                <ns4:SvcLvl>
                    <ns4:Cd>SEPA</ns4:Cd>
                </ns4:SvcLvl>
                <ns4:LclInstrm>
                    <ns4:Cd>INST</ns4:Cd>
                </ns4:LclInstrm>
            </ns4:PmtTpInf>
```

2

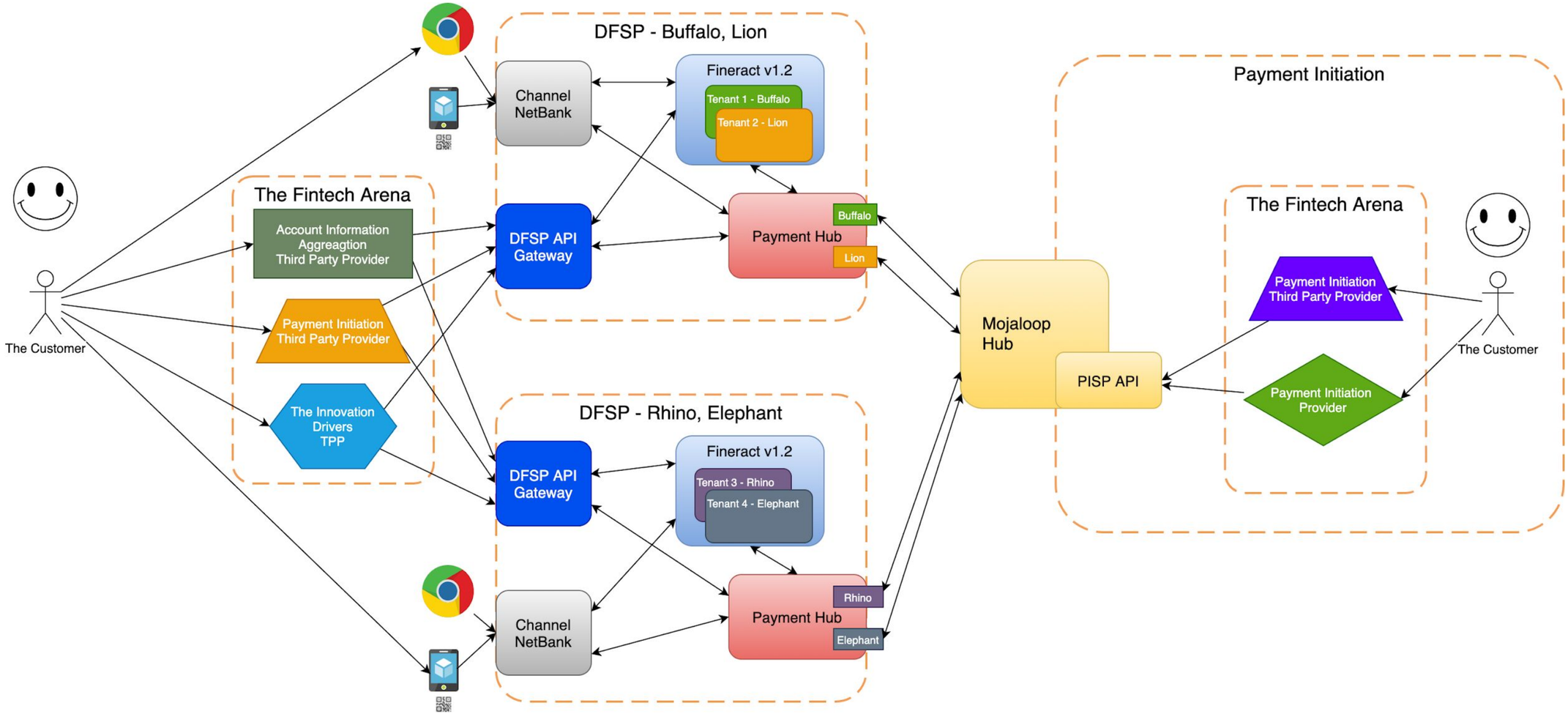
```
                <ns4:CdtrAcct>
                    <ns4:Id>
                        <ns4:IBAN>HU8710700024000000009876543
                    </ns4:Id>
                </ns4:CdtrAcct>
                <ns4:Purp>
                    <ns4:Cd>OTHR</ns4:Cd>
                </ns4:Purp>
                <ns4:RmtInf>
                    <ns4:Ustrd>mjlp:msisdn:27710203999</ns4:Ust
                </ns4:RmtInf>
            </ns4:CdtTrfTxInf>
        </ns4:FIToFICstmrCdtTrf>
    </SendCreditTransferRequest>
```

```
{
  transferId:77e4e8f3-35e0-4819-bfae-f43468ef1
  payerFsp:in03tn05,
  payeeFsp:in02tn03,
  amount:{
    amount:512,
    currency:TZS
  },
  ilpPacket:AQM1MTIgZy50ei5pbjAydG4wMy5N
```

4

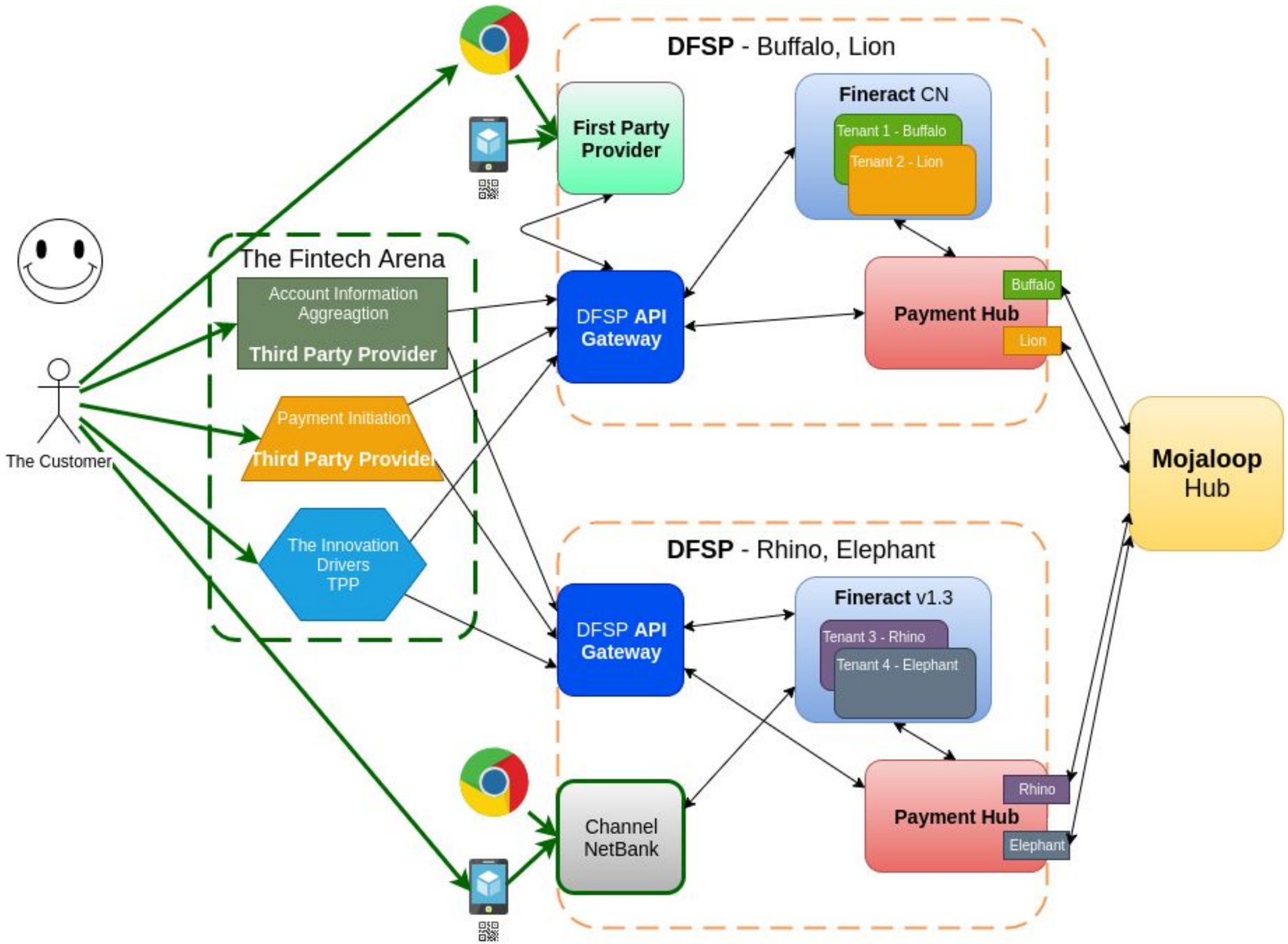
```
{
  transactionId:77e4e8f3-35e0-4819-bfae-f43468ef1186,
  quoteId:fad8d8ec-1aea-489f-88d4-d329a2565340,
  payee:{
    partyIdInfo:{
      partyIdType:MSISDN,
      partyIdentifier:27710203999,
      fspId:in02tn03
    }
  },
  payer:{
    partyIdInfo:{
      partyIdType:MSISDN,
      partyIdentifier:27710305999,
      fspId:in03tn05
    }
  },
  amountType:RECEIVE,
  amount:{
    amount:512,
    currency:TZS
  },
  transactionType:{
    scenario:TRANSFER,
    initiator:PAYER,
    initiatorType:CONSUMER
  }
}
```

Providing Openbanking API to Fintechs



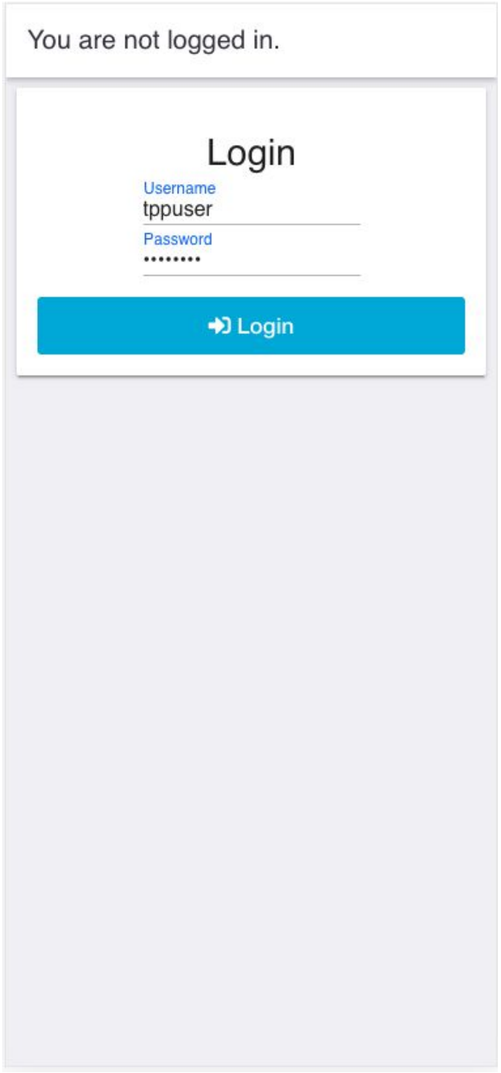
- Supporting the OpenbankingAPIs at DFSP
- Working towards to enable the PISP API implementation

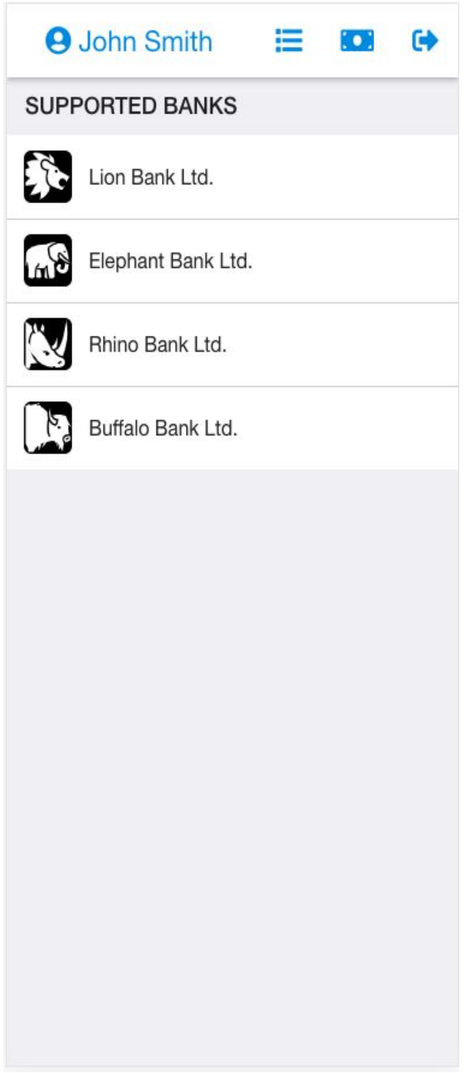
Transaction Initiation thru 3rd Party Fintech via Open Banking API

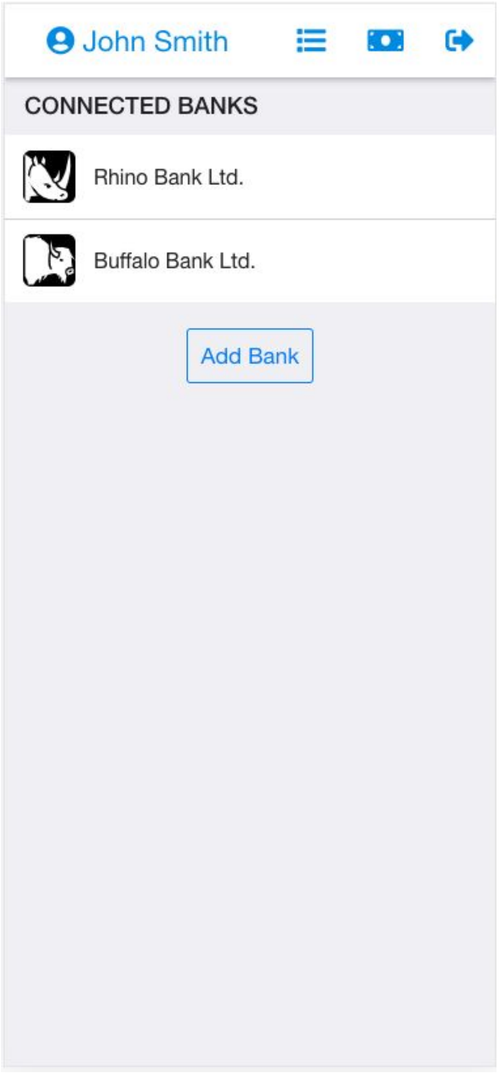


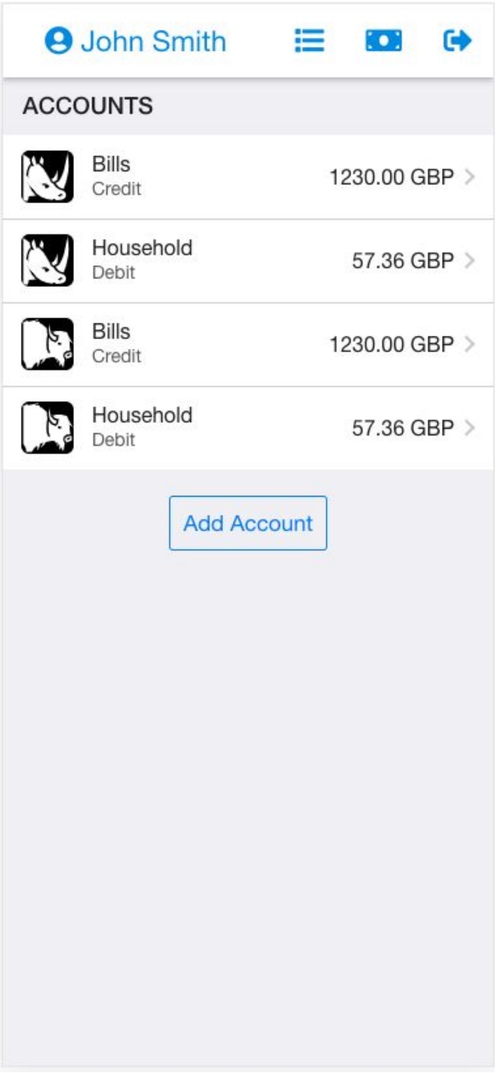
- Enable TPP to access multiple DFSPs, self service onboarding
- Using standardised Open Banking APIs (PSD2, Berlin Group)
- Client and user authentication only once, authorization (OAuth2)
- User gives consent on account, action type, transaction level
- Input validation, fraud monitoring, load balancing, metrics

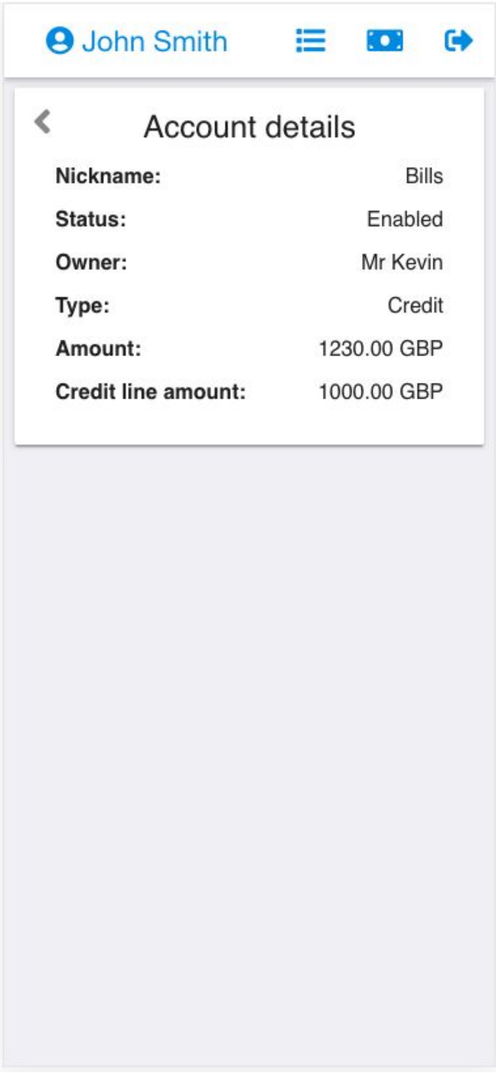
Interaction with the Payment Service User











Additional Use Cases Supported

Integration & Orchestration Layer

- MFI Collection and Disbursement
- Merchant QR Code Payments

Payment Hub as Bulk Pre-Processor

- G2P/Bulk Payment Pre-Processing
- Identity-Driven Payments

Payment Hub as Gateway Layer

- Stablecoins & CBDC

What is CoDi

CoDi is a software platform developed by the Mexican Central Bank (BANXICO) which is used for digital payments in real time using QR and NFC over the existing SPEI Infrastructure which is the Interbank payments platform also developed by BANXICO.

Person or merchants can create payment request for services and products using the smartphone.

The payment request can be on demand or scheduled.

Persons and Merchants are notified in real time about the status of the financial transactions.

Person to Person (P2P)

Payment Request Creation

QR code is created and encrypted. The message is sent to the destination.

Payment Request Reception

Reads and decrypt the QR code for Accepting or Rejecting the Payment Request

Apache Fineract

Financial Institution A

Mifos Payment Hub EE

Central Bank acknowledges the TXs

Apache Fineract

Financial Institution B

Mifos Payment Hub EE

Payment Request/Settlement/Aceptance Messages are sent by the CoDi Infrastructure

Merchants to Persons (B2P)

Payment Request Creation

QR code is created and encrypted. The message is sent to the destination On Demand or Scheduled.

The Phone now is the POS

Reads and decrypt the QR code for Accepting or Rejecting the Payment Request

Payment Request Reception

Apache Fineract

Financial Institution A

Central Bank acknowledges the TXs

Financial Institution B

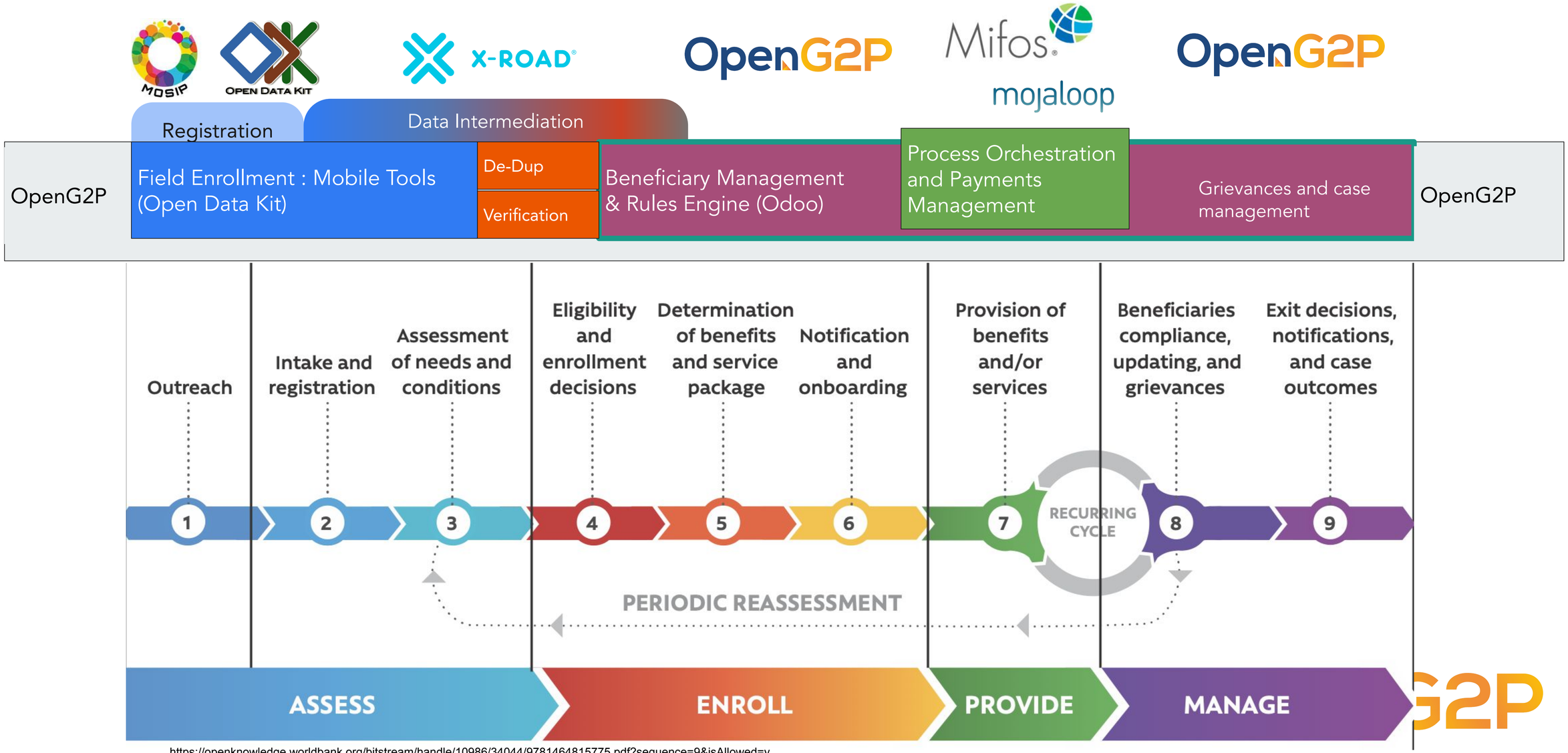
Apache Fineract

Mifos Payment Hub EE

Payment Request/Settlement/Aceptance Messages are sent by the CoDi Infrastructure

Mifos Payment Hub EE

Social Protection - Mapping to SourceBook (World Bank)



https://openknowledge.worldbank.org/bitstream/handle/10986/34044/9781464815775.pdf?sequence=9&isAllowed=y

MOSIP

X-ROAD®

Apache Fineract

Mifos®

mojaloop

Identity

Foundation & Functional ID

+

Data Exchange

Programmatic Sharing of Beneficiary Data

Accounts

Store of Value, Accounts, Wallets & Financial Ledger

+

Orchestration

Bulk Pre-Processing, Payment Integration, Workflow Orchestration

+

Payments

Interoperable Payments, IIPS, Last Mile Delivery

OpenG2P

Transfer System
Built on Financial Inclusion Community of Practice DPGs

OpenG2P

Mapping Systems to Functional Blocks

Enrollment/Assessment | Beneficiary/List Mgmt | Payments Orchestration & Delivery | Reconciliation & Recourse

< ---------- OpenG2P Building Block Architecture ---------->

MOSIP

Identity Provisioning

De-Dup/Verification

Assessment Tools, Case Mgmt

ODK based

odoo OpenG2P ERP, openIMIS, CoreMIS

Enrollment Decision Making & Scoring

Rolls of Beneficiaries & List Management

Beneficiary Management

Lifecycle Management & Workflows

X Road

Information Sharing & Data Intermediation

Mifos

Apache Fineract

mifos/Fineract

Accounts, Transactions, banking features

mifos Payment Hub EE

Payments Engine & Process orchestration

mojaloop Interop-Switch

Vouchers

External Channels

Banks

odoo OpenG2P ERP, openIMIS, CoreMIS

Notifications

Beneficiary Recourse & Complaint Handling

Reconciliation & Monitoring

Reporting & Dashboards

Existing configurable software by Mifos

Existing software

Areas of ongoing focused development & refinement

Roadmap - Next phase of dev

Payments Building Blocks

Disbursement Lists - Odoo

Accounts - Mifos/Fineract

Bulk Processing - Payment Hub EE

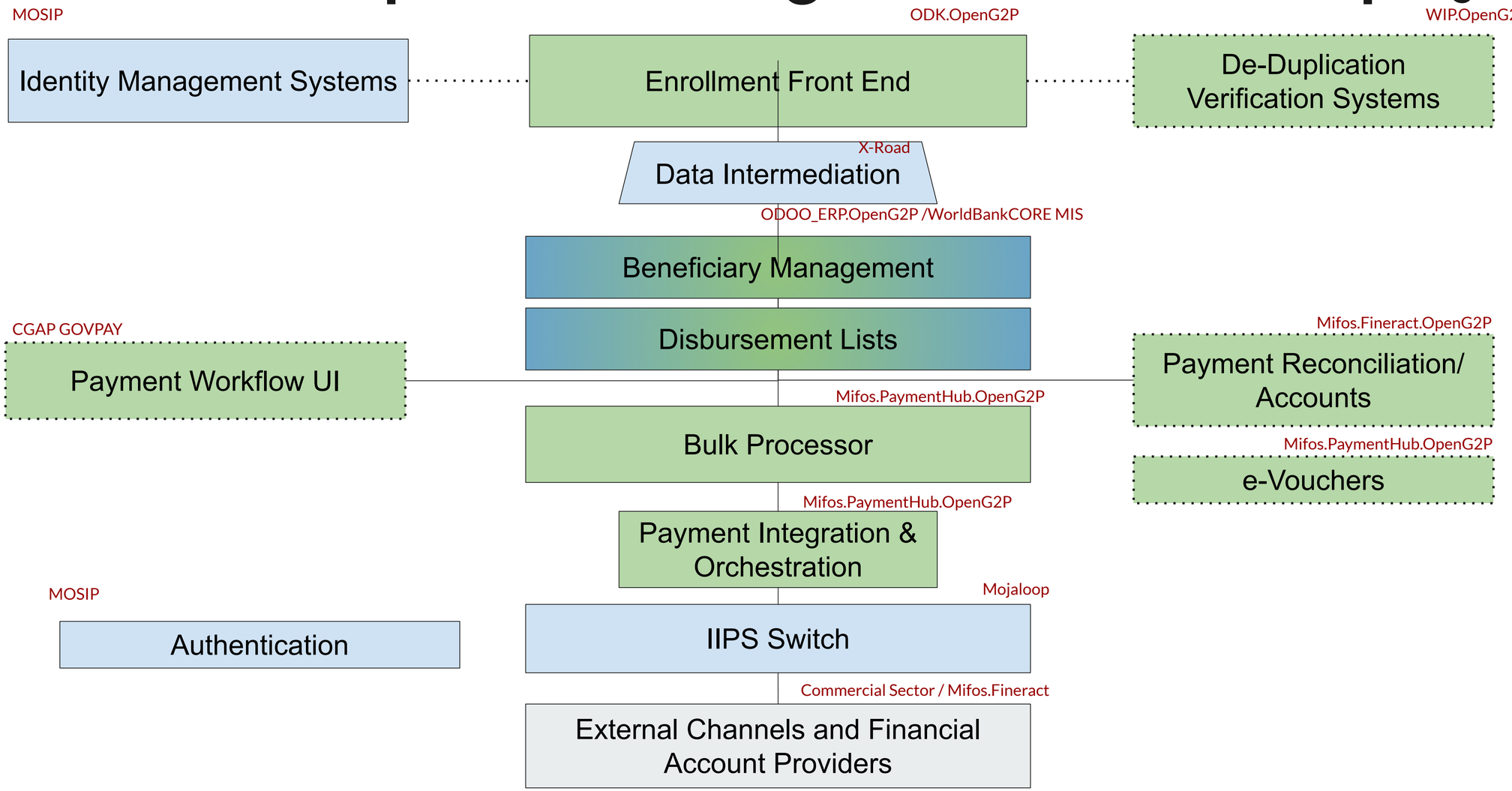
Transmitting Payments - Integration & Orchestration
Payment Hub EE

Interoperability - Mojaloop

Recording Receipts - Mifos/Fineract/Odoo

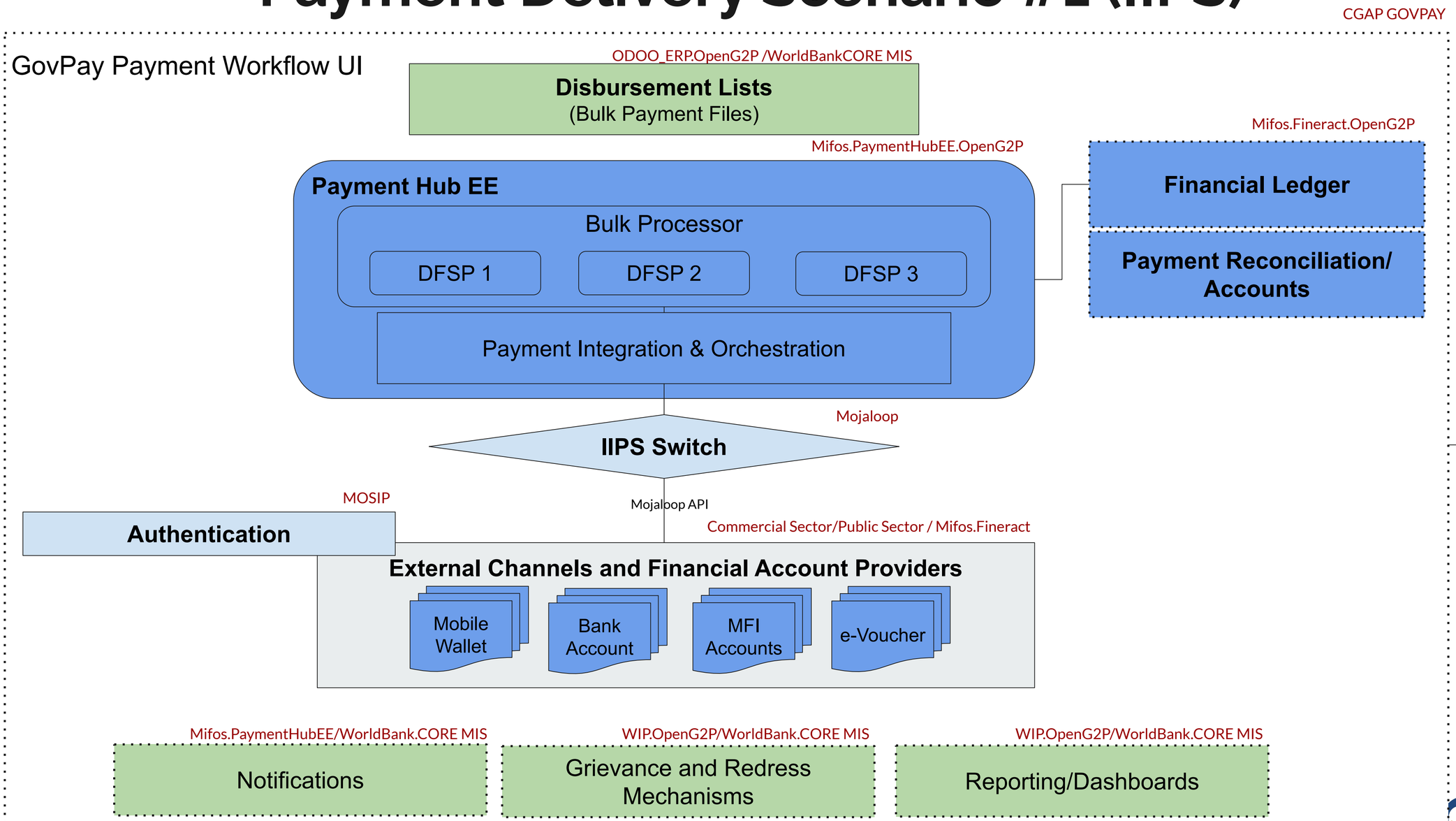
OpenG2P

OpenG2P's integration of different projects*



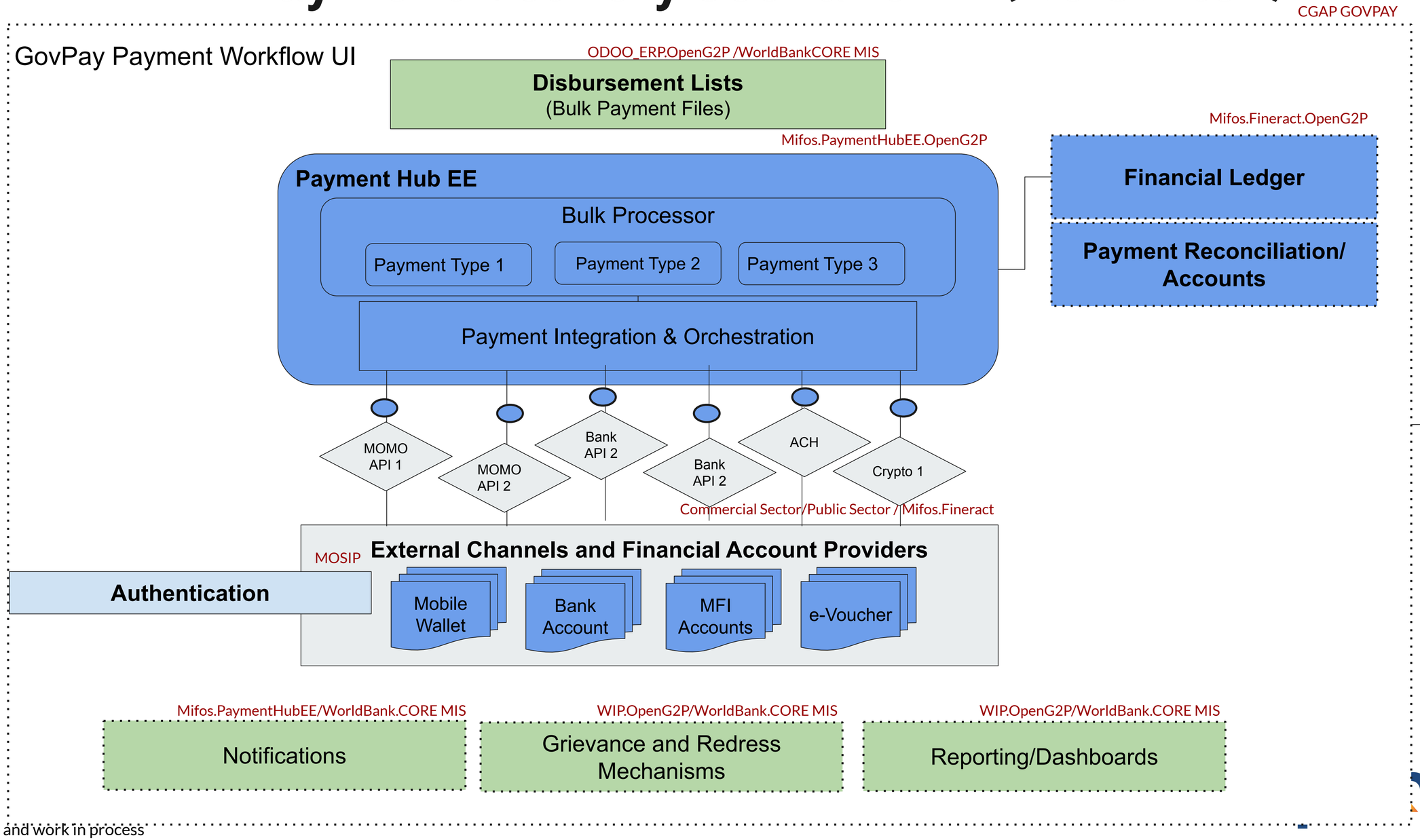
*Planned and work in process

Payment Delivery Scenario #1 (IIPS)



*Planned and work in process

Payment Delivery Scenario #2 (No Switch)

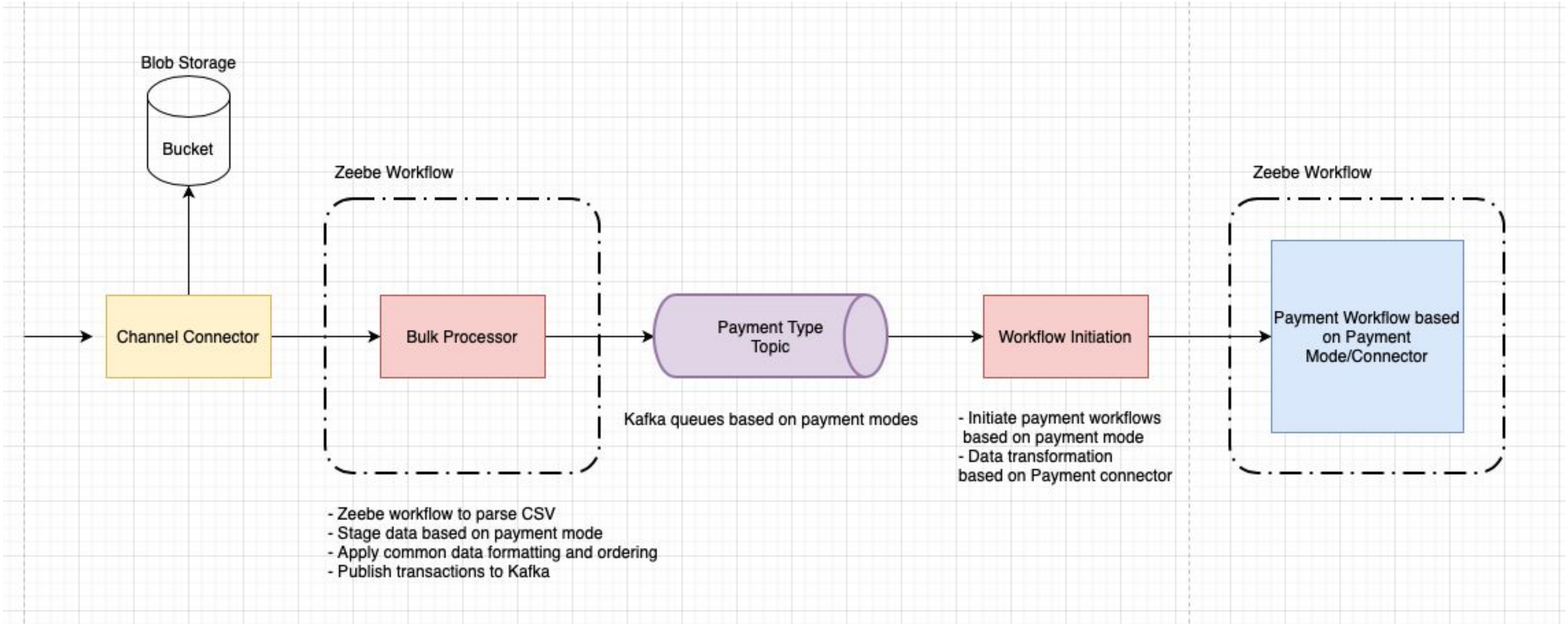


*Planned and work in process

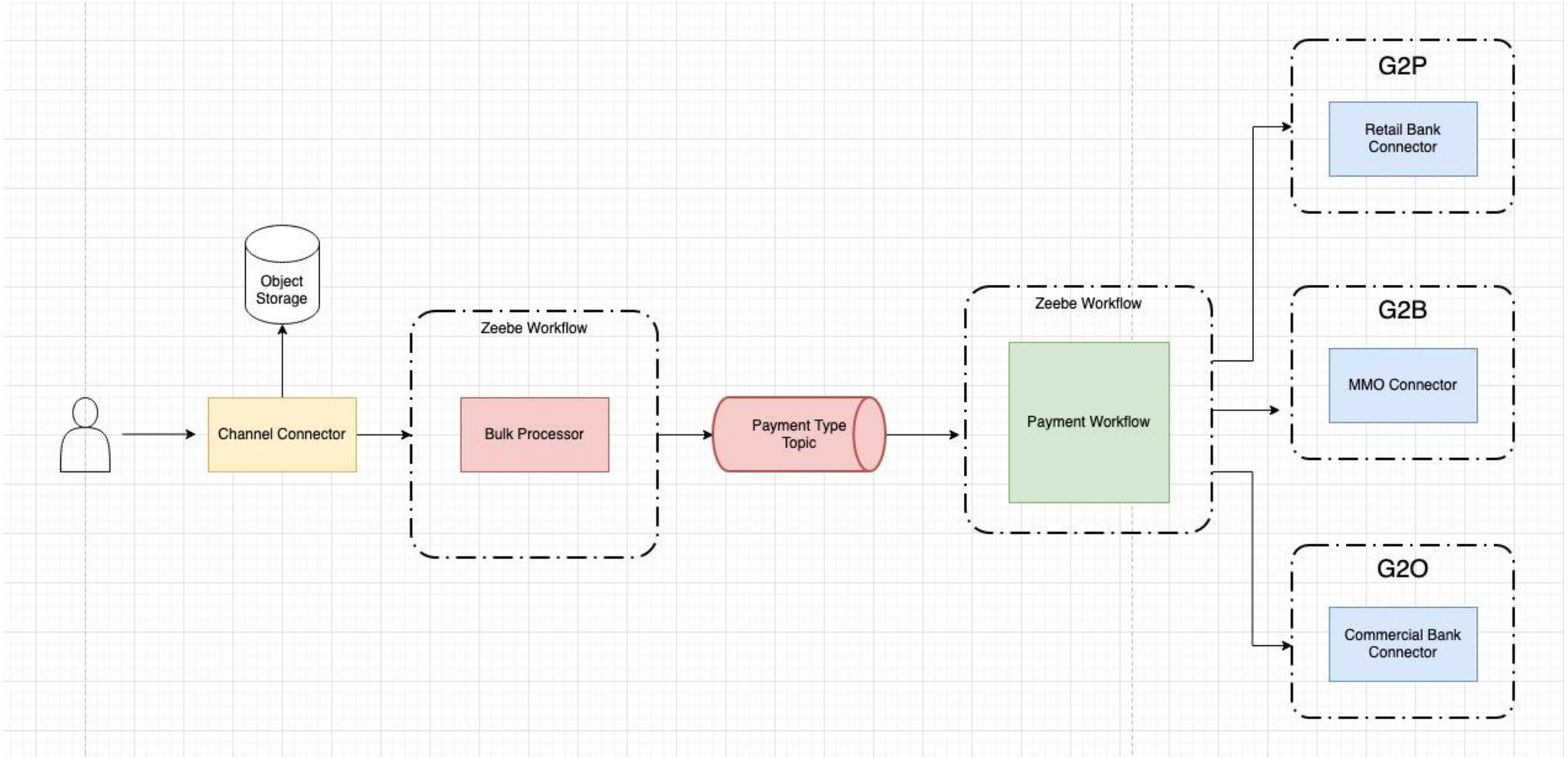
Bulk Payment Pre-Processing Support

- Preprocessing bulk payments
  - Lookup Payee's DFSP ID for the transactions to determine target DFSP/payment type
  - Splitting the incoming bulk into smaller batches per target DFSP/payment type
  - Manage communication with Mojaloop for the batches
  - Aggregate incoming results to provide response to the bank's channel
  -

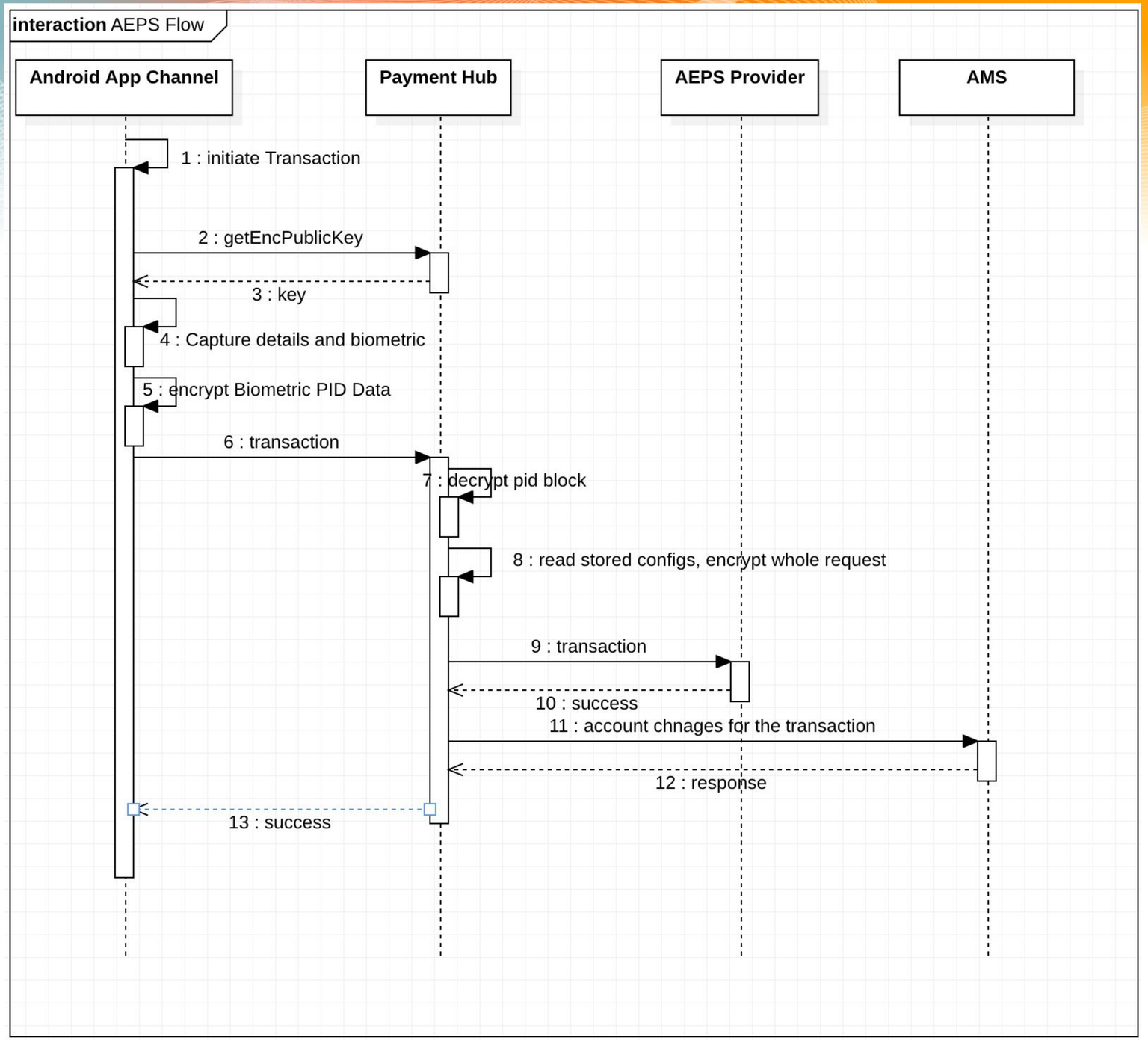
Bulk Processor



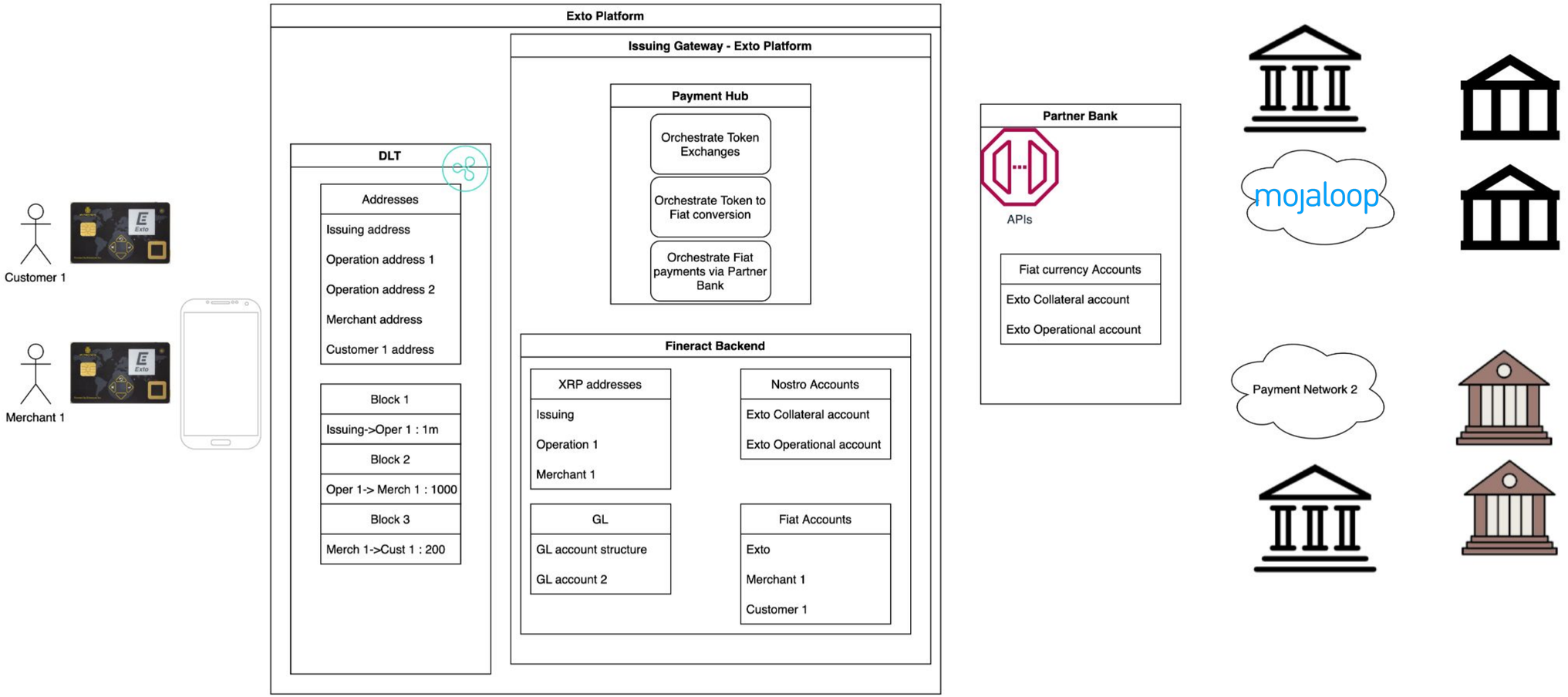
Bulk Processor



Biometric Enabled Payments



Payment Hub as Gateway component



Source: ExtoLab.com, developments in India for MEITY, connected to a private XRP based system. Similar design for Stablecoins and CBDCs.

Roadmap

- ❑ Integrate with FARM service at DFSP level
- ❑ Mojaloop Bulk Payment API Integration
- ❑ Mojaloop PISP API Integration
- ❑ Developer other payment and core banking system connectors
- ❑ Advance other use cases in Africa
- ❑ Stand-in Capabilities

Open for Collaboration

- ❏ Digital Public Good Collaboration around Social Protection & OpenG2P
- ❏ Collaboration with Africanenda to advance IIPS.
- ❏ In discussion with the FARM team around integrating with PH-EE
- ❏ Mifos Lab Environment going live again using Azure Credits for OS
  - ❏ Integration with Mojaloop Community Sandbox
  - ❏ Leverage Sybrin efforts with Microsoft to Optimize
- ❏ Stablecoin & CBDC innovation
- ❏ GSMA MM API and Interoperability Lab Integration
- ❏ Community & Ecosystem Development
  - ❏ Share our knowledge and learnings in enabling upstream contribution model
  - ❏ Continue cross-pollination across partners/integrations from our ecosystem
  - ❏ Sybrin already using Fineract/Mifos for sandboxes

Thank You

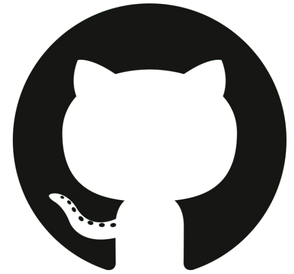
•

Edward Cable
edcable@mifos.org
https://mifos.org

Istvan Molnar
istvan.molnar@dpc.hu
https://dpc.hu

mojaloop foundation

github.com/openMF
github.com/apache/fineract
https://fineract.apache.org



Browse the Docs:
https://mifos.gitbook.io/docs/payment-hub-ee

Explore the Code:
https://github.com/openMF?q=ph-ee

Discuss on Slack: https://bit.ly/3eMoVS1

Request Access to the Lab:
https://mifos.gitbook.io/docs/payment-hub-ee/overview/lab-environment

Backups

Key architectural considerations of Payment Hub EE 1/2

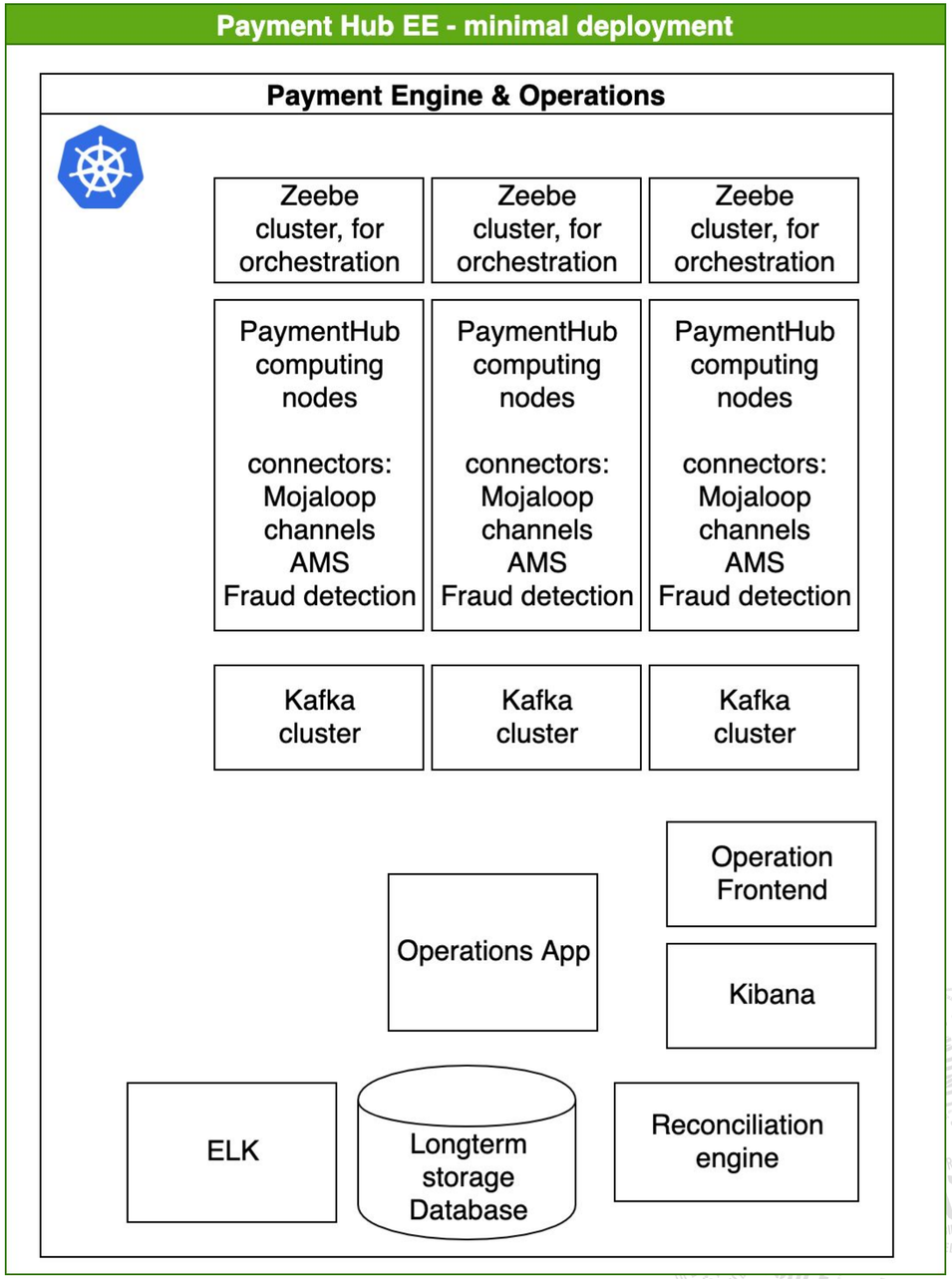
- Running on **on-premises** and **cloud** infrastructures, utilizing Kubernetes
- Separate **real-time engine** and **operations backend** systems
- The **Payment Realtime Engine** is **self contained, highly available and fault tolerant**, can handle complete transaction flows
- **Orchestrating** the microservices are desired
  - handling timeouts
  - correlations of asynchronous events
  - handling error and exception scenarios with the necessary compensations
  - overview of in-flight transactions
- A Payment Realtime Engine manages the **state machine using a microservice orchestration** layer
  - Using Raft for consensus and log replication
  - Using RocksDB for key-value store of states
  - Does not use external NoSQL or Relational DB, which hinders scalability

Key architectural considerations of Payment Hub EE 2/2

- **One engine is bounded to a single data center**, for high performance, low latency and minimizing issues from network failure scenarios
- **Multiple realtime engines could run parallel and independent**
  - Protect against complete site failure
  - 24x7, 99.99+% availability operations require version upgrades, certificate changes, hardware replacement, ... without interruption of the service
  - If payment network supports the notion of independent engines at a single DFSP, than routing could happen at the Switch, otherwise, the Payment Hub realtime engines could hand over the callbacks amongst each-other internally
- Systems for **backoffice operations**, including long term storage database, audit logging, monitoring, reconciliation - **detached in a separate cluster**
  - These components **could be offline without any payment service interruption** and data loss - the realtime engines Kafka layer keeps the records
  - “Operations systems” are not performance critical
  - Backoffice user access allowed only to these components, not the realtime engines

Deployment model - minimal

- Minimal deployment
  - Single kubernetes cluster to contain all the necessary components
  - Fault tolerance provided by the clustered components
  - Stretched installation across data centers possible, but not ideal
- Full scale deployment
  - Multiple independent payment engines (a single engine is collocated for performance), enabling complete version upgrades without service interruptions
  - Running in different data centers on independent network connections (high availability, fault tolerant even in case of disaster scenarios)
  - Partitioning the load across the engines



What is Zeebe and why to use it?

Open source microservice orchestration engine, implemented in **highly scalable, highly performant** and **fault tolerant** way using state of the art, proven technology building blocks

- Using Raft protocol for consensus and replication
- Using RocksDB, an embeddable high performance key-value store, for its persistency
- All process instances are auditable, and audit records can be sent to Elasticsearch or Kafka
- Using an efficient, high performance binary protocol (gRPC) for the clients to connect and receive tasks for execution

Key features of Zeebe

Workflow definition	BPMN 2.0, YAML - provides integrators with an easy to use tool to customize workflows
Visual workflow representation	Yes, BPMN workflows are both executable and have visual representation
State storage for active workflow instances	Stored on the machines running Zeebe using embedded RocksDB (no external storage required)
Scalability	Horizontally scalable via partitions (no DB bottleneck)
Fault tolerance	Yes, via replication factor (3 node, 1 can fail, or 5 node and 2 can fail)
Supported programming languages	Ships with Java and Go clients, plus NodeJS, C# and Ruby clients available, using gRPC for the clients
Historic workflow data	Can be streamed to storage systems via exporters (Elasticsearch and Apache Kafka is available). Complete auditable log.

OUR Solution - an End to End Open Source Stack for Digital Financial Services

We have integrated a complete demonstration of ISO 20022 messages through Mojaloop via the “Payment Hub EE” - using open source software.

Entire stack is Open Source and API-driven

Bridge & Orchestration Engine
Enables seamless connection to any payment system & maps messages from ISO 20022 to Mojaloop format.

Core Banking Platform
Cloud-native open source core banking platform for account & portfolio mgmt.

Reference Mobile Apps
Mobile banking, wallet, and fintech apps that can be white-labeled & extended.

ISO 20022

Payment Hub EE

THE LEVEL ONE PROJECT

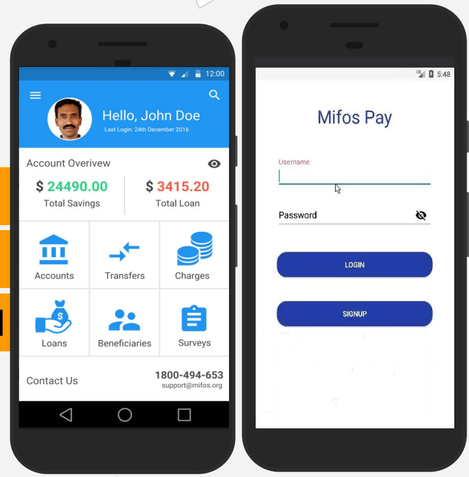
mojaloop

Apache Fineract

Open Banking APIs

PISP APIs

GSMA Mobile Money API



- Design concept for ubiquitous, inclusive, open loop, instant, low-cost payment systems

- Full payment hub for any Financial Service Provider
- Maps minimal set of ISO 20022 data to Mojaloop API

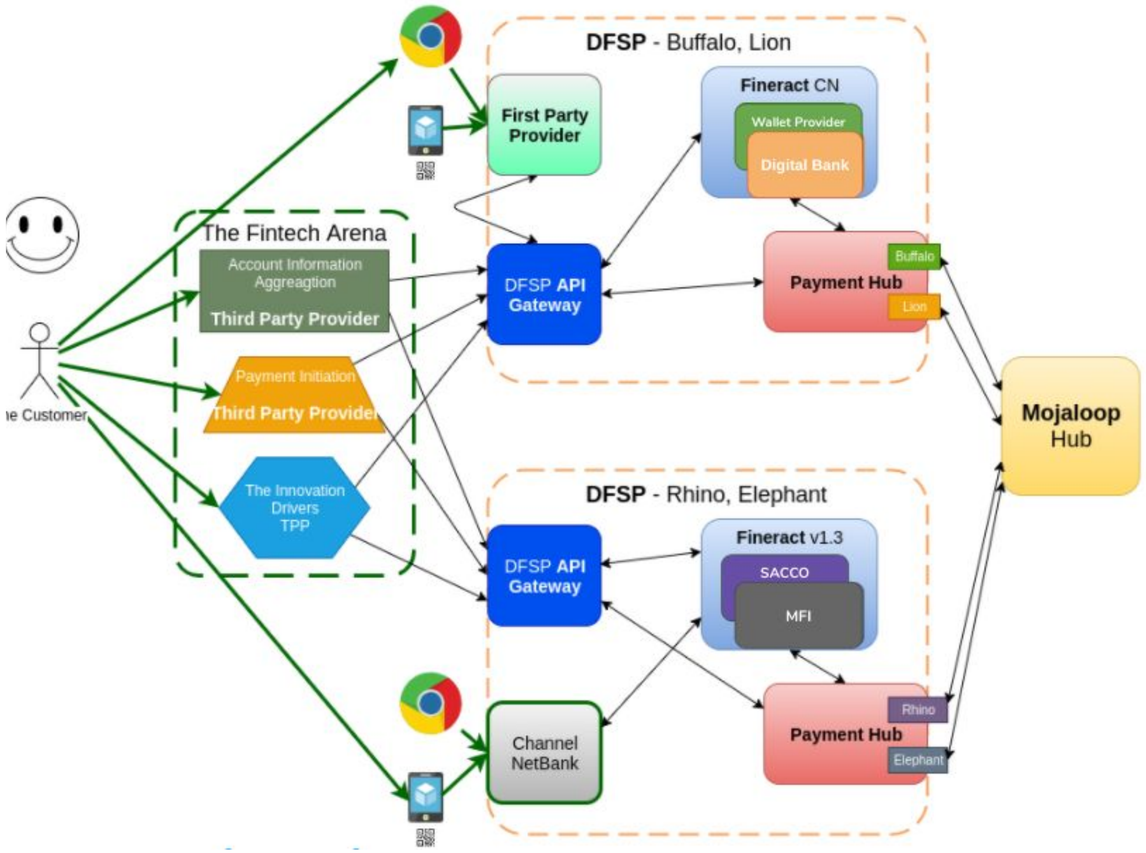
- Unlocks last-mile delivery by enabling interoperability & participation of non-banks
- Open Mojaloop API that is evolving to ISO 20022 standard focusing on the minimal essential data components of the standard.

- Scalable Cloud Account opening and hosting
- Provides modern CB infrastructure to leverage rich ISO 20022 data and deliver adjacencies and value-added services on top of rails.

- End customers benefit from richer ISO 20022 data.
- Enables participation of third parties and fintechs

Rules | Rails | Accounts | Apps

Connect the Mifos Lab with the Mojalab



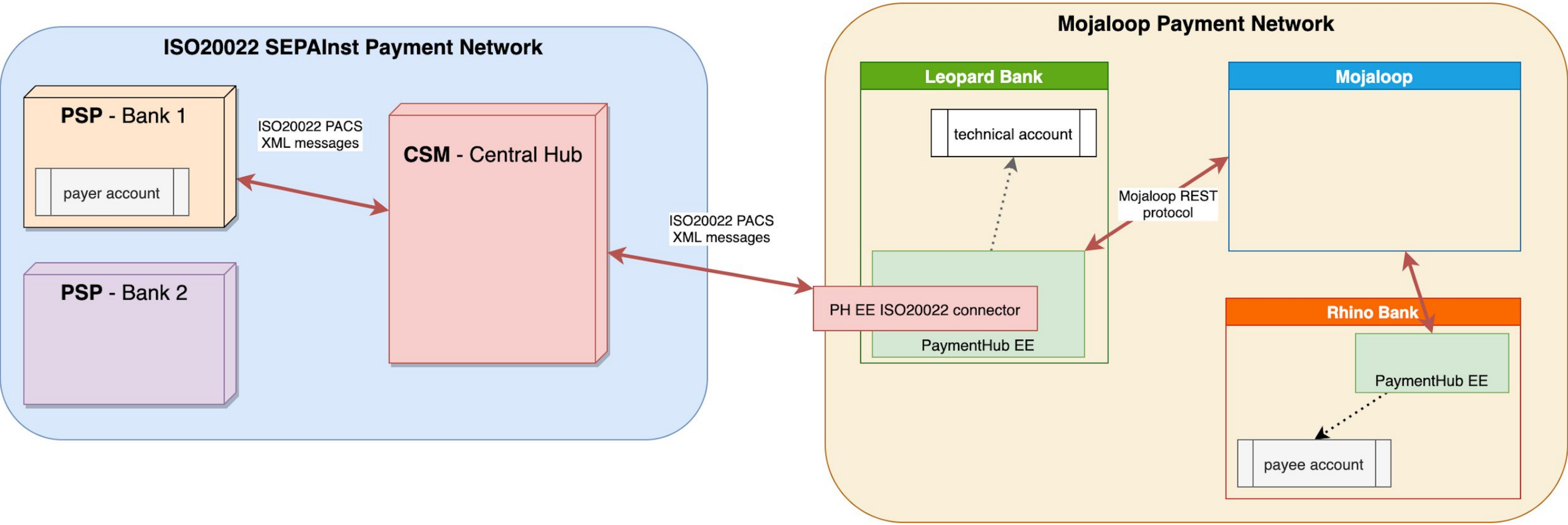
Audience	API Layer	OS Toolset
Fintechs	- **Open Banking API Layer** - **GSMA Mobile Money API (in progress)**	➢ Mobile Wallet App ➢ Mobile Banking App ➢ Online Banking App ➢ Open Banking Fintech App
DFSPs Banks, MFIs, SACCO, MMOS	**Mifos/Fineract API Layer** - Identity & KYC - Wallet/Account Management - Loan & Savings Management - Accounting & Ledger - Reporting	➢ API-Driven Open Source Core Banking Platform ➢ Angular Web UI for staff ➢ Android Mobile UI for staff
Hub Operators, Regulators	**Mojaloop API Layer** - Peer to Peer - Merchant Proximity Payment - Merchant Request to Pay - Bulk Payment/Transfer (upcoming)	➢ Payment Hub EE ○ Operations UI ○ BPMN Workflow Engine

- ❏ Provide reference DFSP UIs
- ❏ Provide reference customer apps - mobile wallet, PISP/fintech app
- ❏ Connect Fineract-based systems like Musoni via Payment Hub

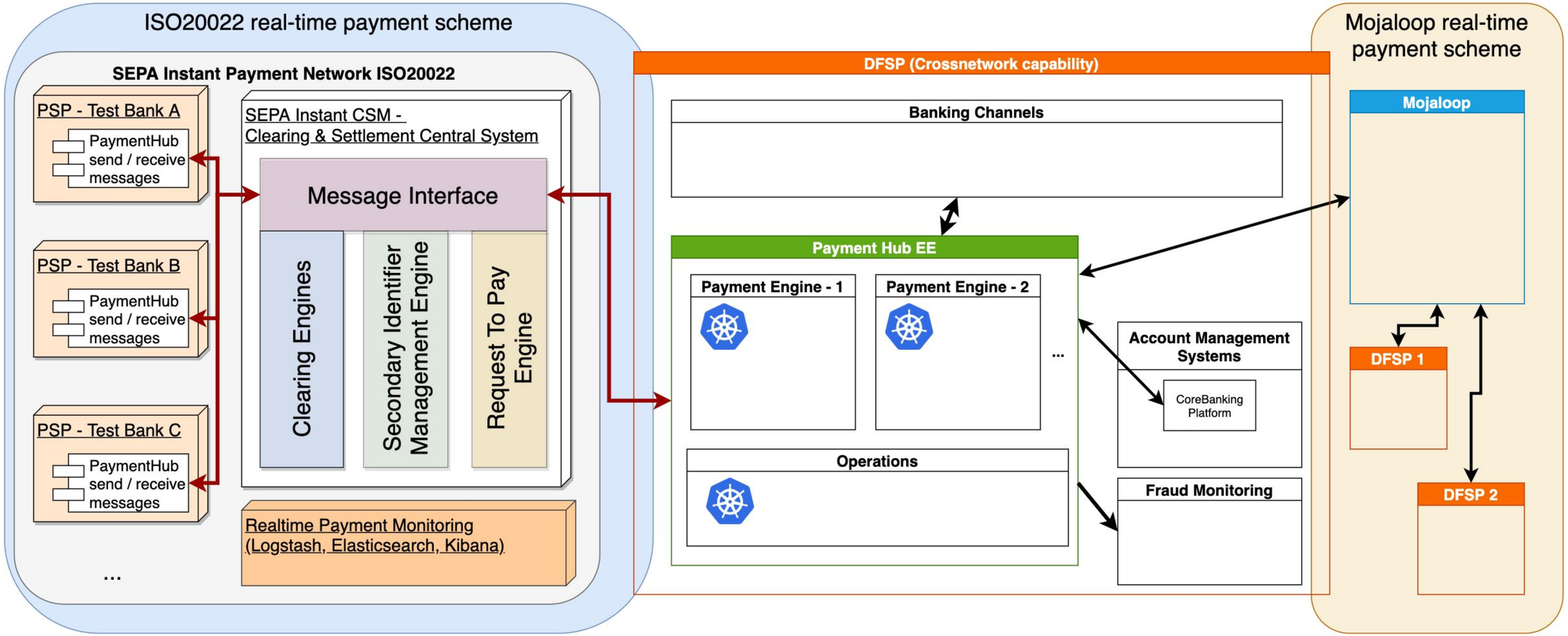
- ❏ Support future DFS Lab hackathons
- ❏ Collaborate with other DPGs like MOSIP & OpenG2P to demonstrate G2P Use Cases
- ❏ Provide open source tooling for regulatory sandbox efforts

mojaloop foundation

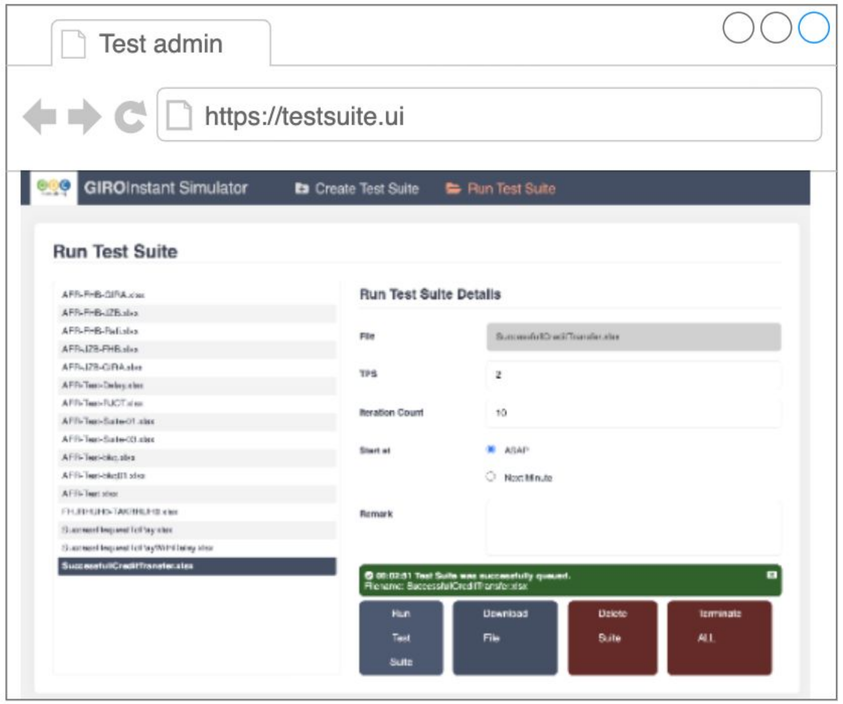
Crossnetwork - ISO20022 (Payments Clearing and Settlement - pacs.008, pacs.002) - Mojaloop

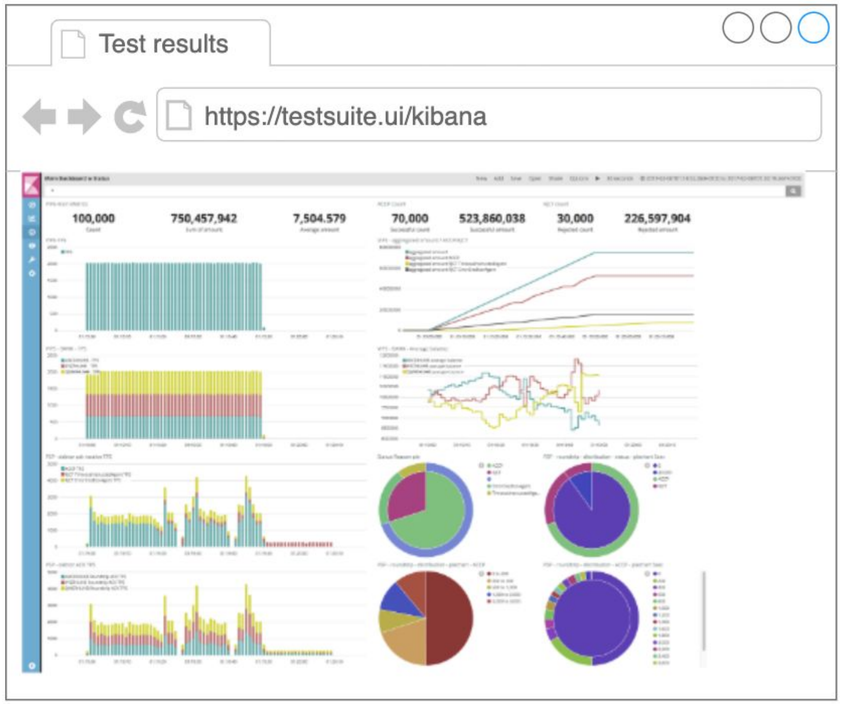


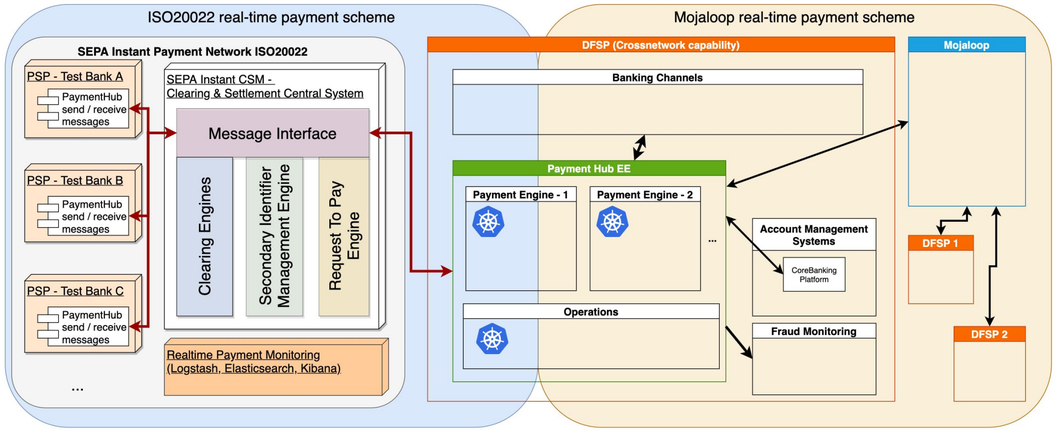
Crossnetwork - ISO20022 (Payments Clearing and Settlement - pacs.008, pacs.002) - Mojaloop



Crossnetwork - ISO20022 - Testing





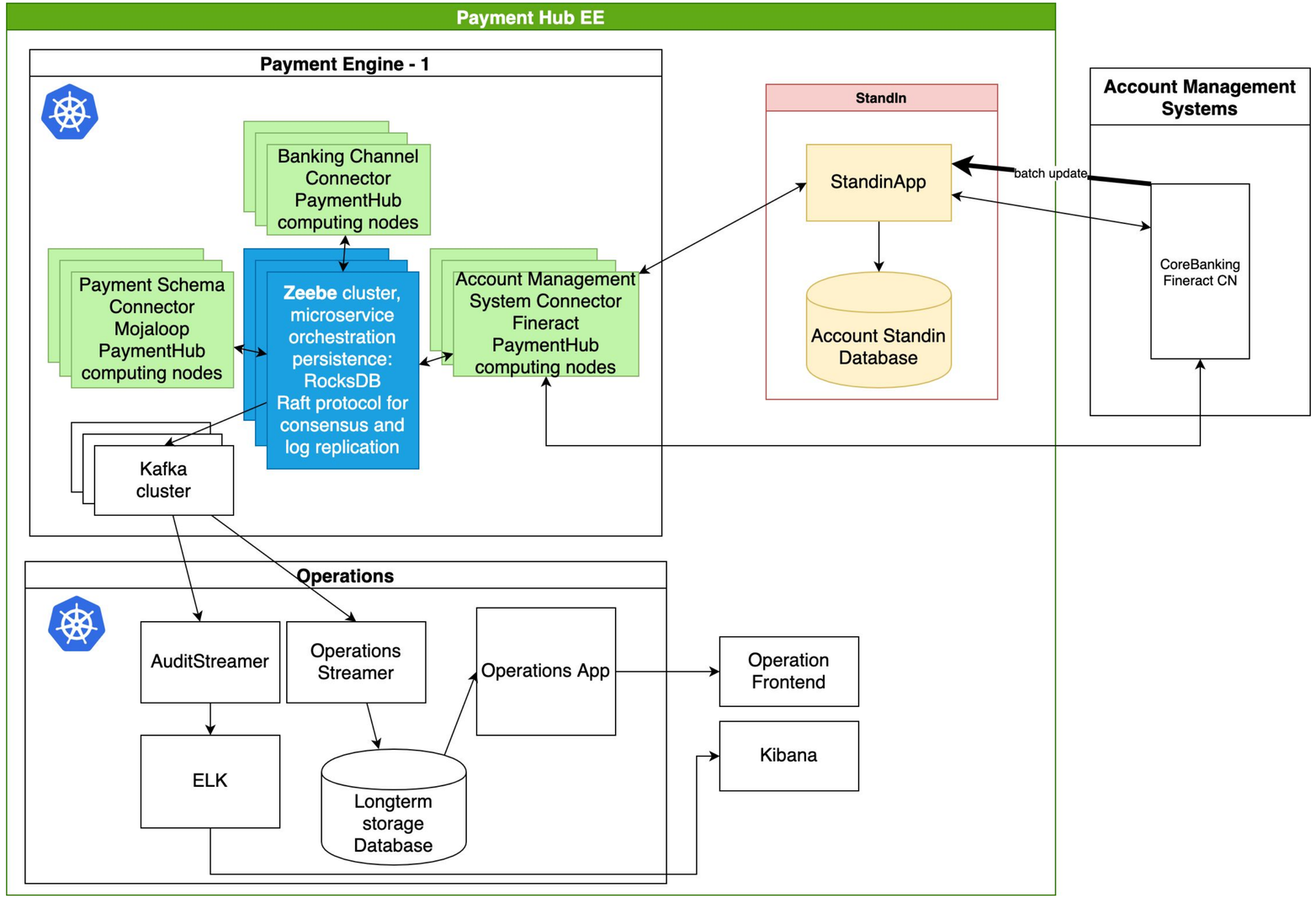


Stand-in

Integrated into the Payment Hub EE solution a stand-in system could provide functionality in case the Account Management System is not available.

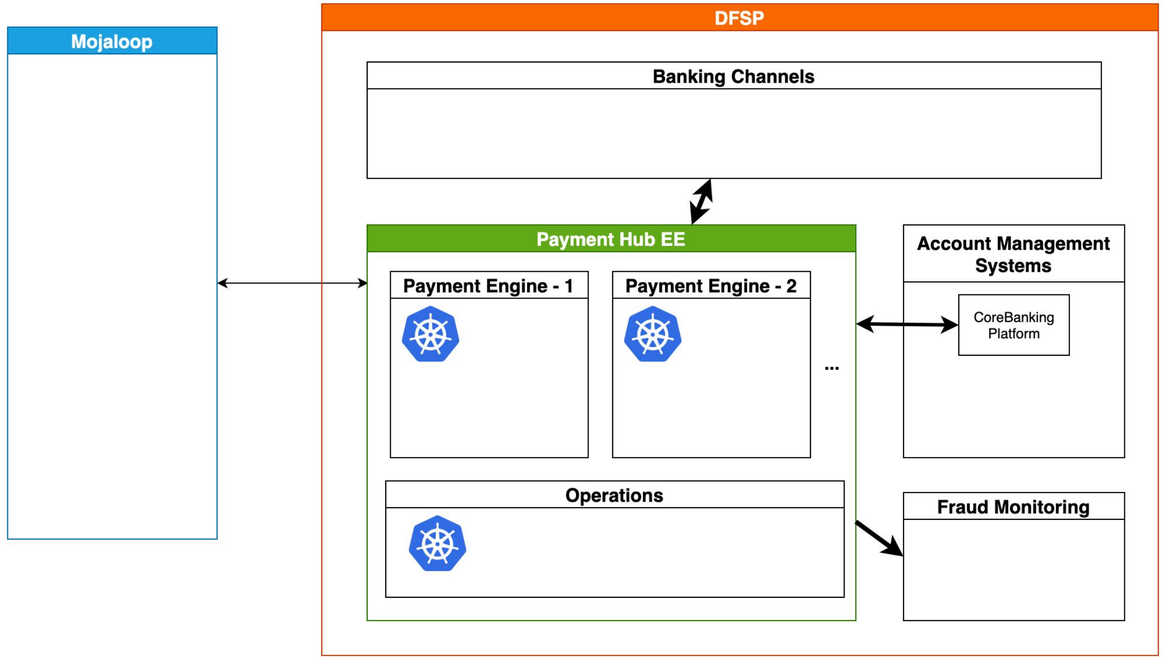
MFI's systems are often not 24x7.

- only incoming transactions - simple solution, requires only synchronizing the valid account and party ids
- both incoming and outgoing transactions - requires more complex solution to minimize risk of overdraft



DFSP Level Fraud Detection - Regulatory Fraud Detection

- The payment process could support the DFSP level Fraud monitoring tools
  - to stop transactions or alert operators
- There are requirements to keep the on-us transactions inside the DFSP
- PHEE can be the component to provide information to regulators with the required amount of granularity of on-us transaction details (considering GDPR and similar regulations)
  - send all transactional information as individual reporting messages
  - aggregate on-us traffic data for a required period (minutes to days)

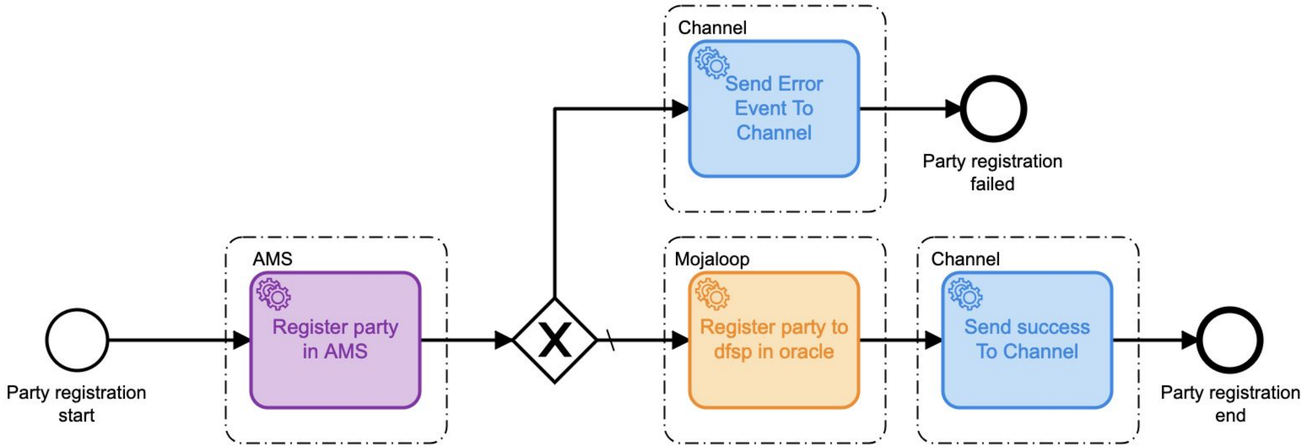


Examples of Use Case Workflows

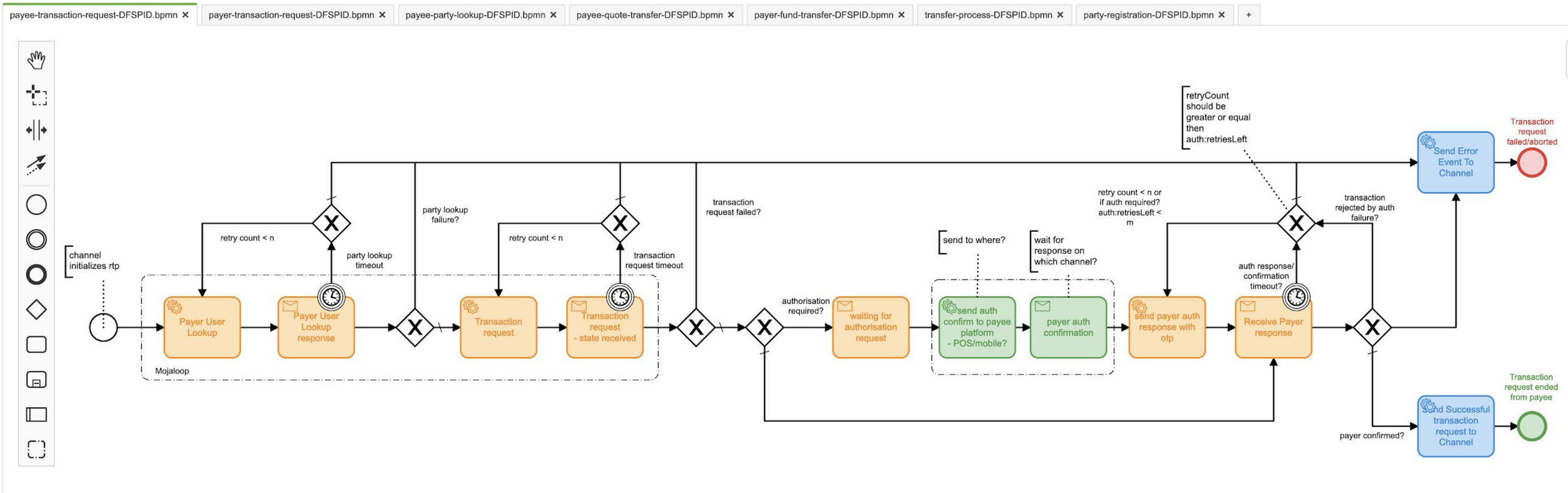
Using Zeebe Modeler for process orchestration at heart of Payment Hub EE

Party Identifier Registration

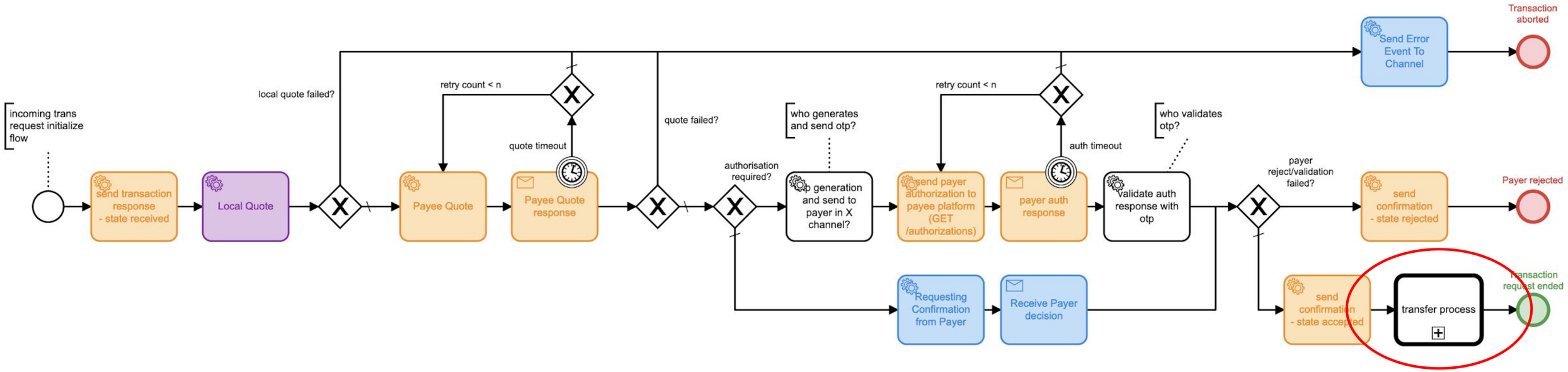
- Initiate the association of a party identifier (MSISDN) with an account at DFSP
- Register identifier in the DFSP systems (in the Account Management System in our environment)
- Manage the registration process with error handling at the “Oracle”



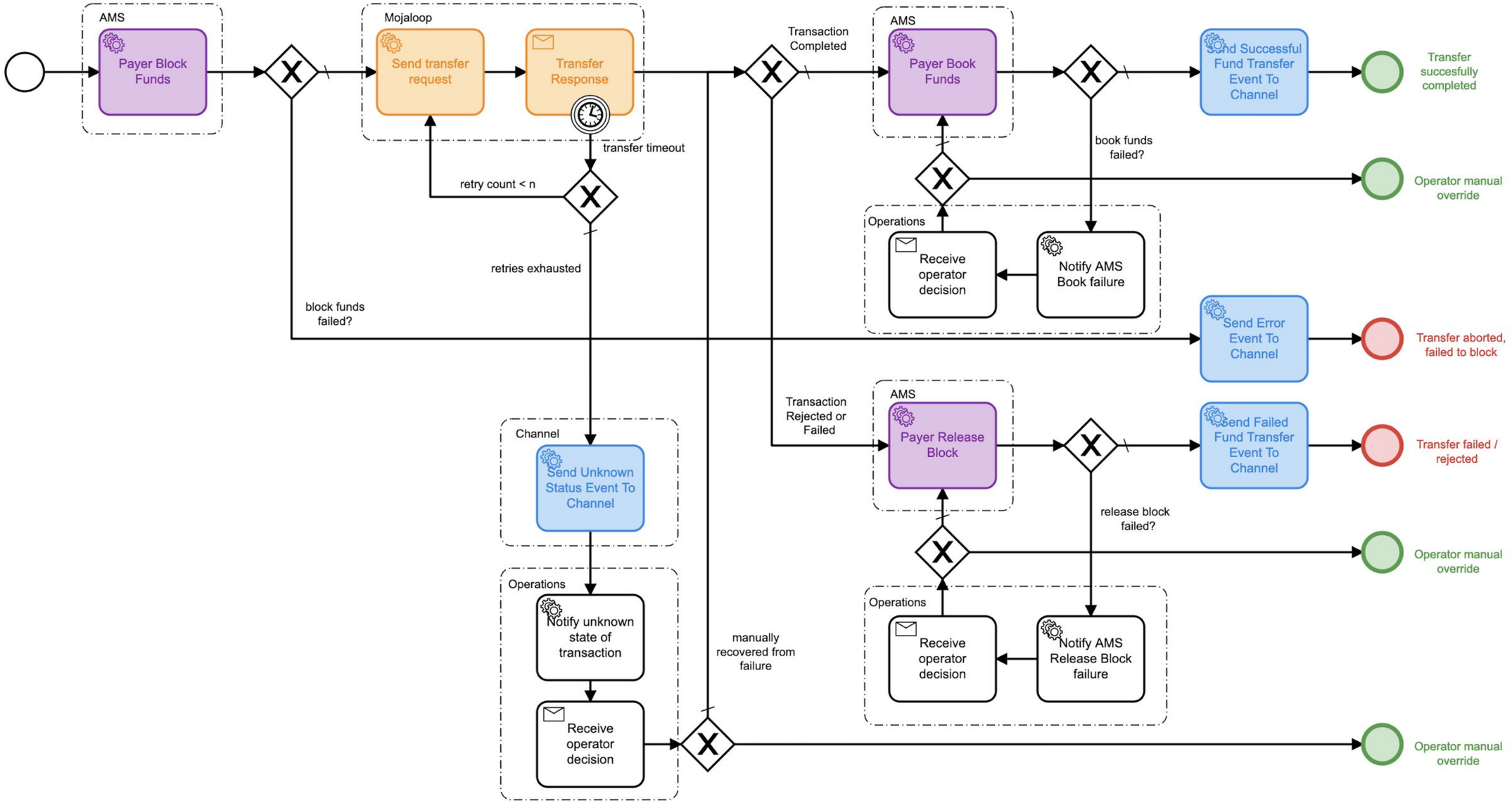
Payee Initiated flows - Request To Pay



Accepting request to pay at Payer



Payer Fund Transfer



Operational Control Center for DFSP actions

Authentication and authorization with privileges for different actions

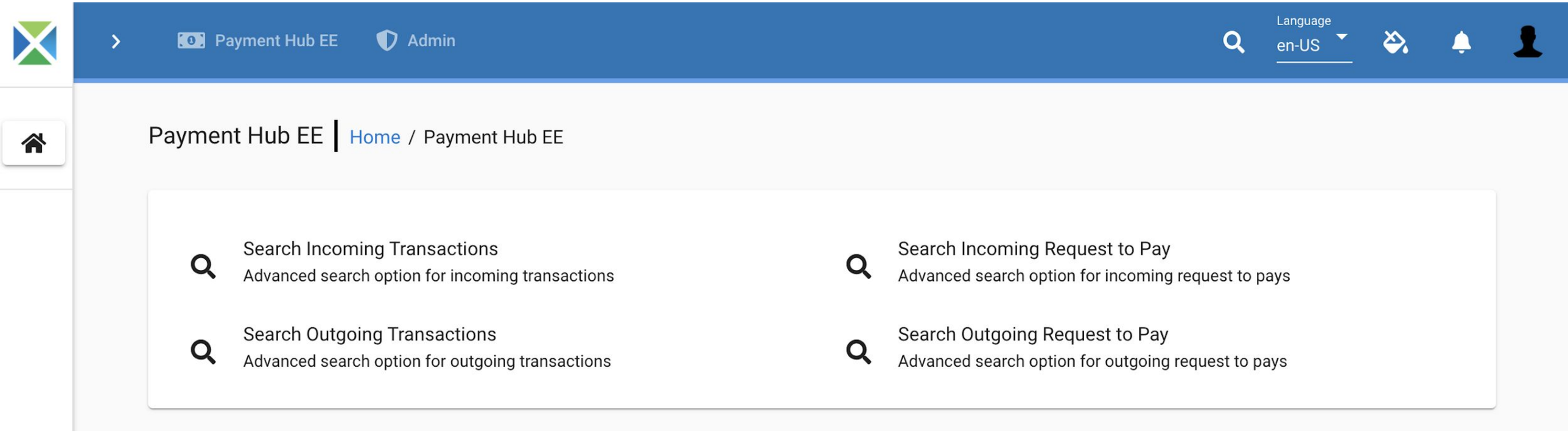
Complete segregation of tenants, separate databases and users

Search and detailed view of transactions

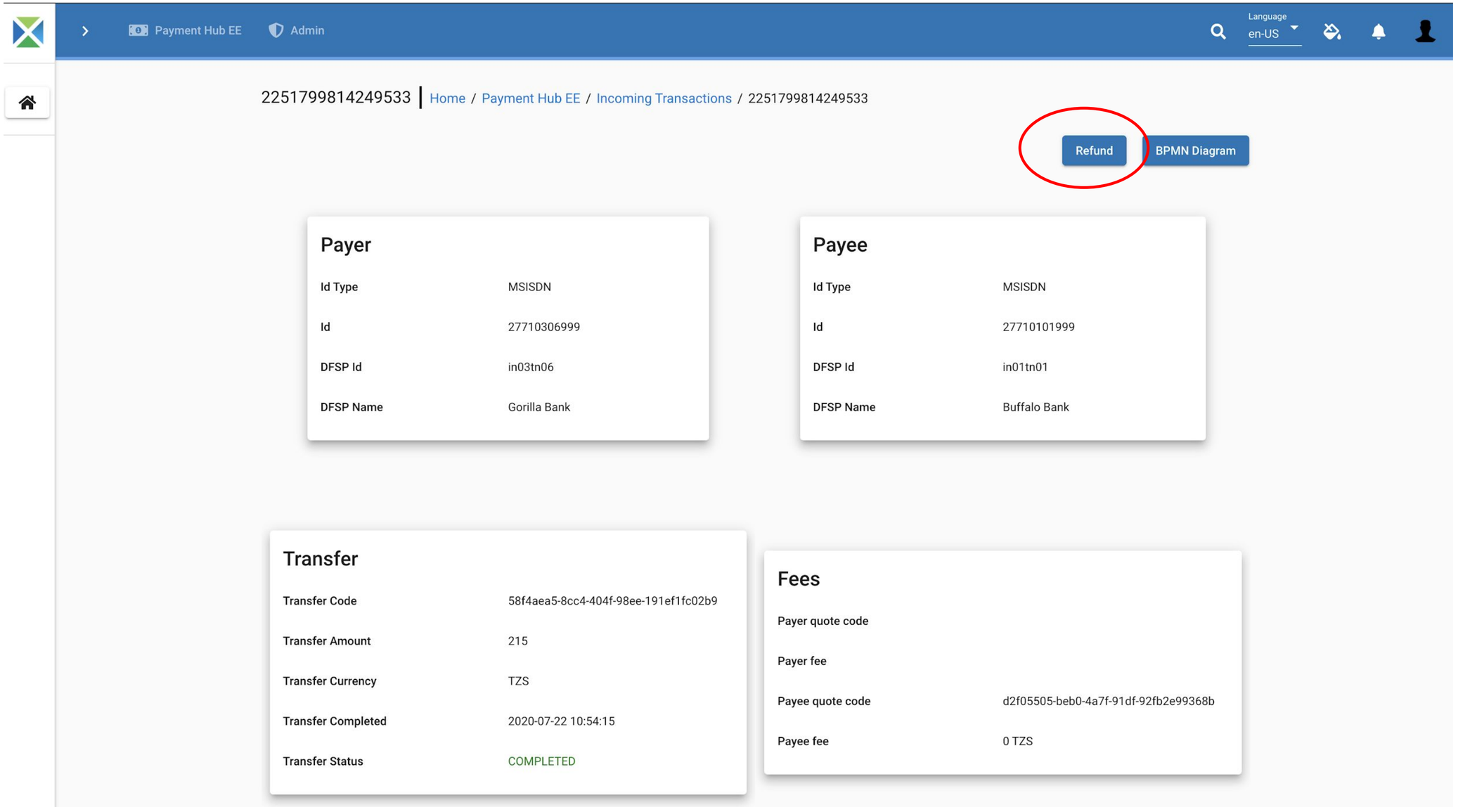
Refund capability for privileged operators on successful incoming transfers

After multiple automated attempts, transactions can be handed over to operations to retry or resolve manually

Search capability for the different flows



Incoming transfers with Refund capability



Outgoing Transfer - The Refund