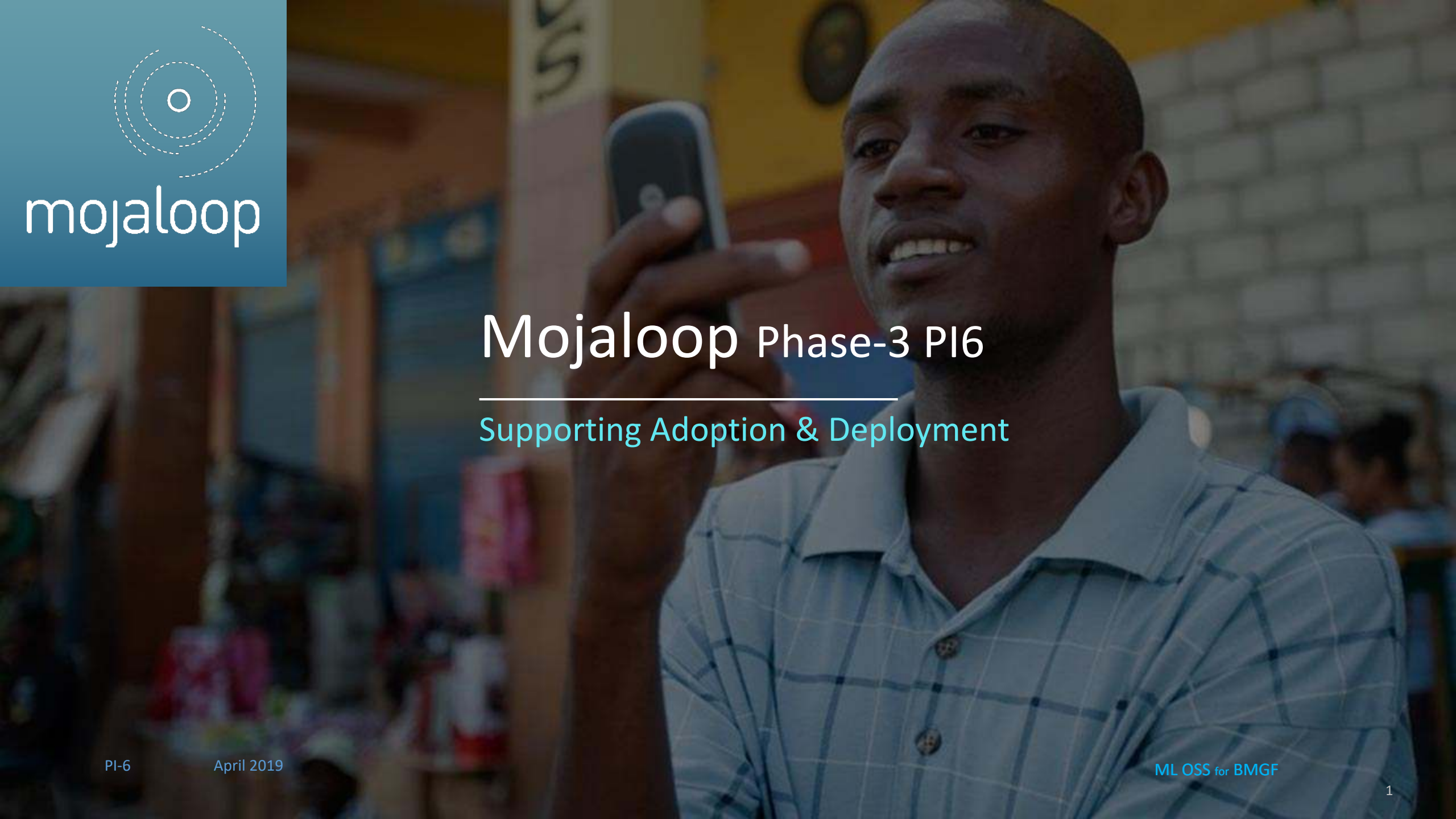


Mojaloop Phase-3 PI6

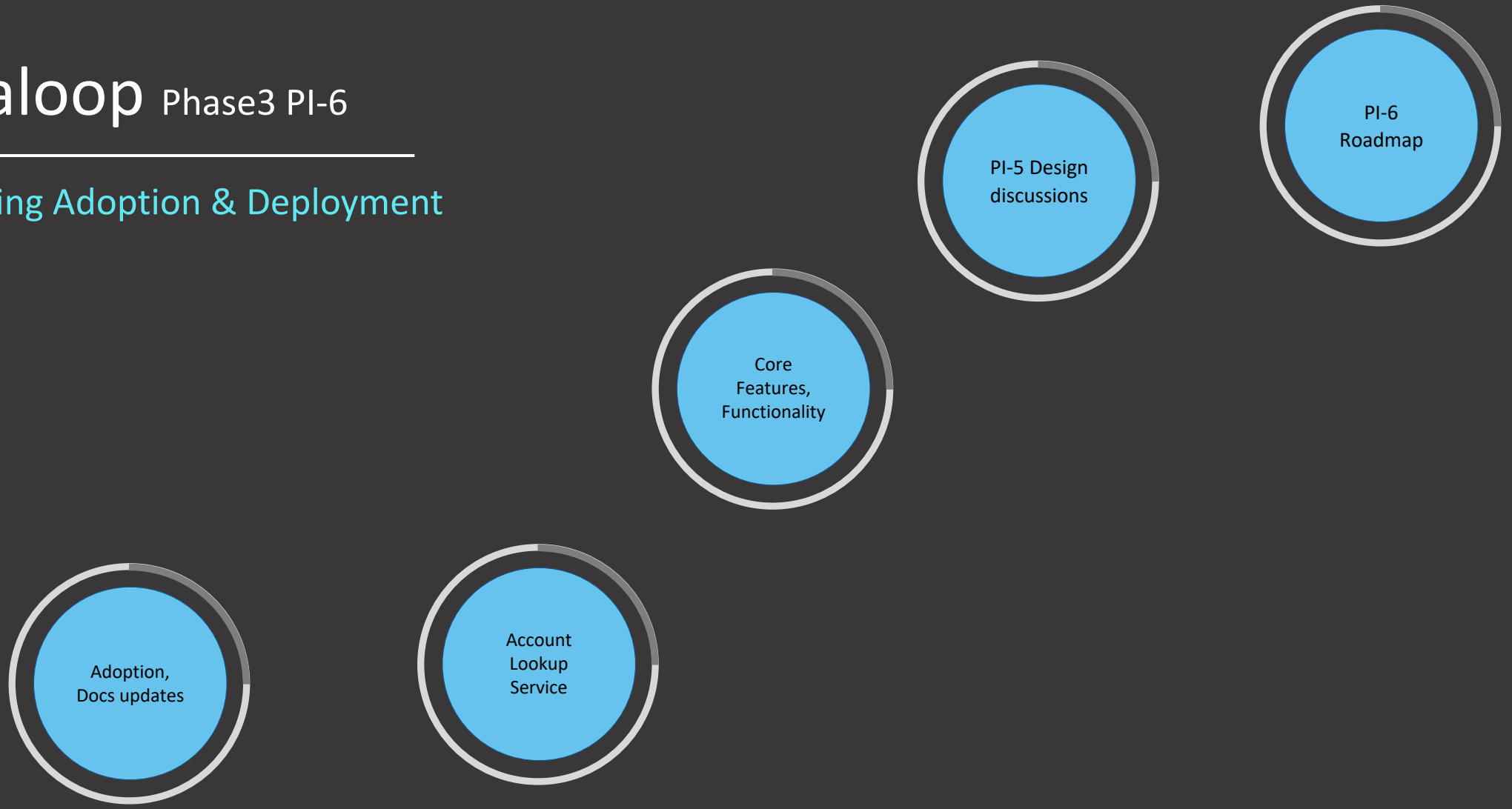
Supporting Adoption & Deployment

PI-6 April 2019

ML OSS for BMGF

Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



16th – 18th April 2019

mojaloop

PI-5 Introduction and Summary

Supporting Adoption & Deployment

PI-6 Agenda - 1

1. Community Updates
    a. Documentation
    b. Adoption Updates
2. PI-5 Account Lookup Service (ALS) Update
    a. ALS Design
    b. ALS implementation

PI-6 Agenda - 2

3. PI-5 Core Features and Functionality updates
    a. Release Mechanism
    b. QA Framework
    c. Integration Tests
    d. Error end-points
    e. Node Upgrade
    f. Bug Fixes
    g. Validation Enhancements
    h. Community Support
    i. Code Enhancements
    j. Metrics Enhancements
    k. Deployment Updates

PI-6 Agenda - 3

4. Topic / Design discussions
    a. Design Authority Proposal
    b. Bulk Payments Design
    c. Settlements Updates and Discussion (Day 2)
    d. Fraud Workstream and Plans for the next PI (Day 2)
5. PI-6 RoadMap
    a. Topics (Day 3)
    b. Priorities (Day 3)

Mojaloop PIs Overview

Timeline	Summary
Phase-1	Level One Project • Reference Implementation • 6 Program Increments (PIs) (2016 - 17)
Phase-2	Road To Productionization: Phase-2 (2018) • PI – 1 (Feb - April) • PI - 2 (April - June) • PI - 3 (June - August) • PI – 3.5 (September) • PI – 4 (November-December 2018): Performance, Settlements, CEP, QA Framework, Operational Monitoring, Managed backlog for Phase-3
Phase-3	Supporting Adoption & Deployment – (2019 Jan - June) • ***PI-5 (Feb – mid-April): Cross-border/network, Account lookup, Comprehensive QA Framework, Streamlined CI, Release process, Support for error endpoints, Documentation, Node Upgrade, Bug Fixes & Community support, Bulk Transfers Design*** • PI-6: Merchant Payments, Error-Event handling frameworks, PoC for Bulk payments, Gateway, Fraud & AML, Settlements API

PI-6 Mojaloop OSS Community

Collaboration

a. Documentation (Cross-Lake, Fintech-Inversiones, Modusbox)

b. Quality Assurance (Mowali, Modusbox)

c. Account Lookup Service (Mowali, Tips, Coil, Modusbox)

d. Scrum-of-scrums (Wider community)

e. Design Authority Channel

Switch Functionality – Mojaloop End-points (PI4)

Mojaloop v1.0 – API Specification

Transfers
[●] POST - Prepare
[●] PUT - Response
[●] PUT – Error
  [●] Outgoing
  [●] Incoming
[●] GET - Query

Parties
[●] GET - Request
[●] PUT - Response
[o] PUT - Error

Quotes
[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[o] GET - Query

Participants
[●] POST - Create
[●] PUT - Response
[o] POST - Bulk Create
[o] PUT - Error
[o] DEL - Delete

Transactions
[o] PUT - Response
[o] GET - Query

TransactionRequests
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

Authorizations
[o] GET - Request
[o] PUT - Response
[o] PUT - Error

BulkTransfers
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

BulkQuotes
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

Key
[●] Fully implemented
[●] Legacy Code
[●] Partially implemented
[●] Not implemented
[o] Out of Scope for PI4

Switch Functionality – Mojaloop End-points (PI5)

Mojaloop v1.0 – API Specification

Transfers
[●] POST - Prepare
[●] PUT - Response
[●] **PUT – Error**
  [●] Outgoing
  [●] **Incoming**
[●] **GET - Query**

Parties
[●] **GET - Request**
[●] **PUT - Response**
[●] **PUT - Error**

Quotes
[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[o] GET - Query

Participants
[●] **POST - Create**
[●] **PUT - Response**
[●] **POST - Bulk Create**
[●] **PUT - Error**
[●] **DEL - Delete**

Transactions
[o] PUT - Response
[o] GET - Query

TransactionRequests
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

Authorizations
[o] GET - Request
[o] PUT - Response
[o] PUT - Error

BulkTransfers
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

BulkQuotes
[o] POST - Request
[o] PUT - Response
[o] PUT - Error
[o] GET - Query

Key
[●] Fully implemented
[●] Legacy Code
[●] Partially implemented
[●] Not implemented
[o] Out of Scope for PI5

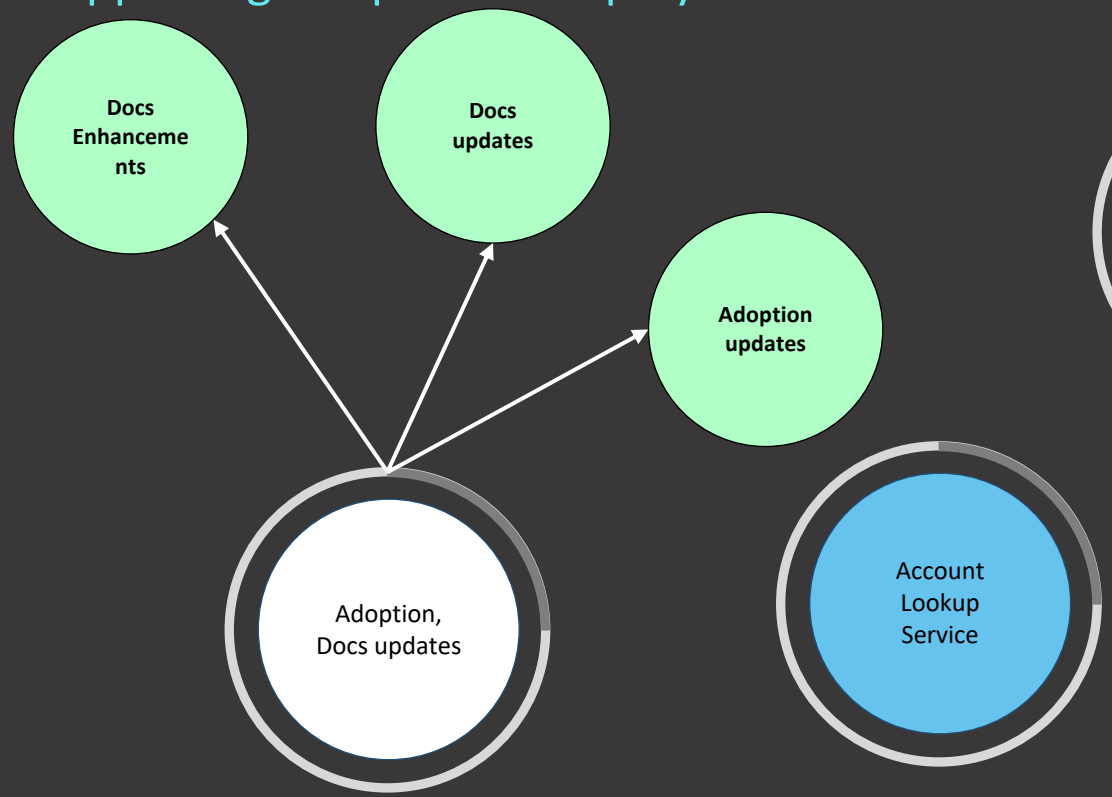
● Interim:
- subId not supported currently
- no validations applied to “update” operations
- full design for Participants POST – Create is pending

PI-5 Documentation, Adoption Updates

Supporting Adoption & Deployment

Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



PI-5 Design discussions

PI-6 Roadmap

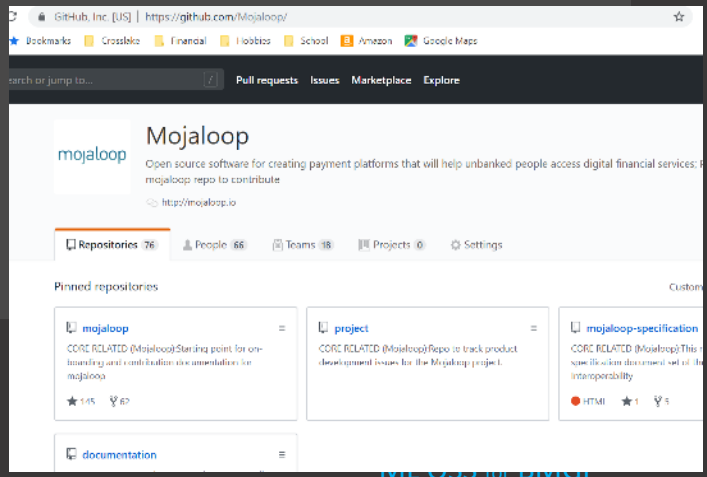
Core Features, Functionality

Account Lookup Service

16th – 18th April 2019

Documentation Overview

- Documentation and Onboarding is critical to any Open Source Project
- Mojaloop.io
  - Provides first entry point into the system
  - Overview of Use Cases – updated this PI
- GitHub
  - Contains project documentation, diagrams and code
  - Open to the public for consumption and updates
  - Challenge for organizing documentation flows



Documentation – GitBooks Overview

GitBook

Ref: http://gitbook.com
Ref: https://en.wikipedia.org/wiki/Markdown

What is Gitbooks?

An open-source open documentation framework where teams can document everything from products, to APIs and internal knowledge-bases based on open-standards with community driven plugins.

Why Gitbooks?

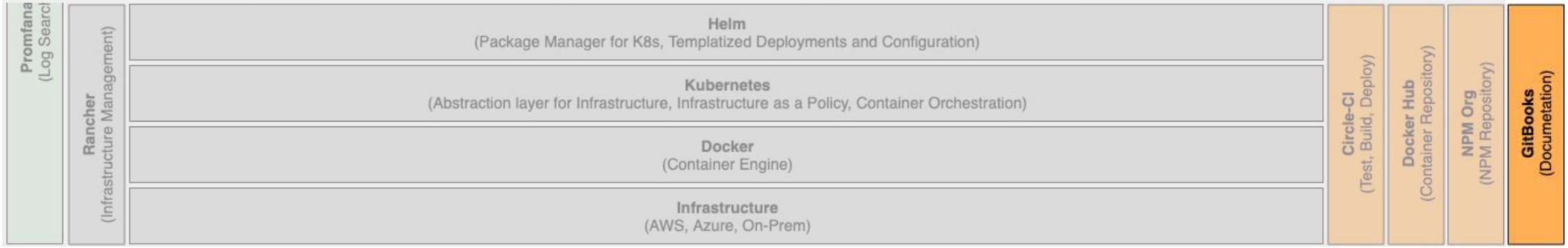
Markdown
Lightweight markup language with plain text formatting syntax supporting standard HTML, and CSS.

Imbed Generated Content
Embed generated sequence diagrams, openapi/swagger docs, etc.

Search
Find what you are looking for.

Plugins
Community plugins for generating content (e.g. plantuml, openapi/swagger docs), providing integration to Github, Slack, etc and themes (e.g. ToCs, Navigation, etc)

Cli
Gitbook-cli to build static-content, with support for local testing. Also supports auto-build sense when changes are made locally when testing.



Documentation - Mojaloop

Quickly submit content changes via Pull-Requests by the click of a button

Hosted through **gh-pages**

GitHub

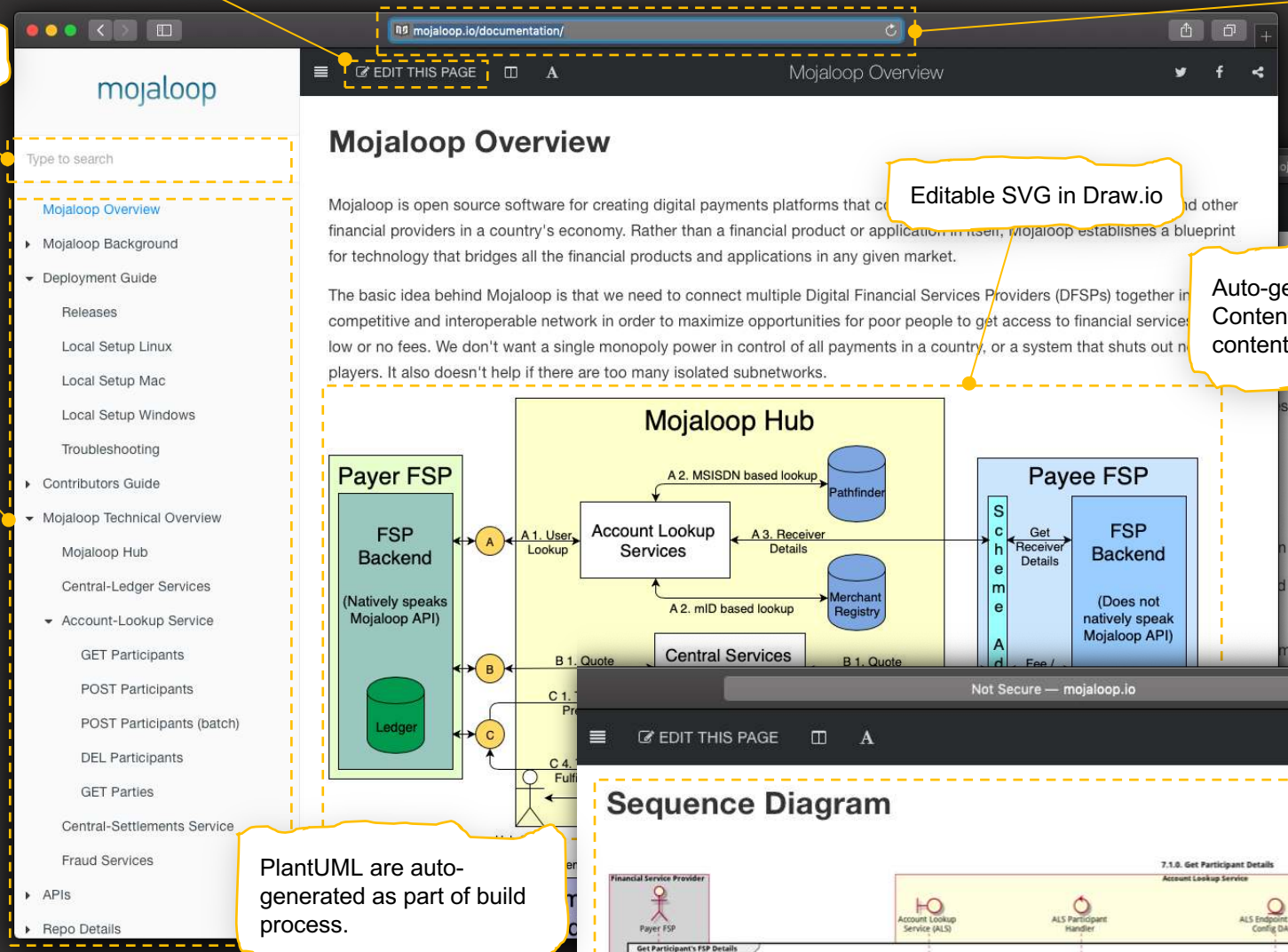
Search capability

Collapsible navigation

Editable SVG in Draw.io

Auto-generated Table of Contents based on page content headers.

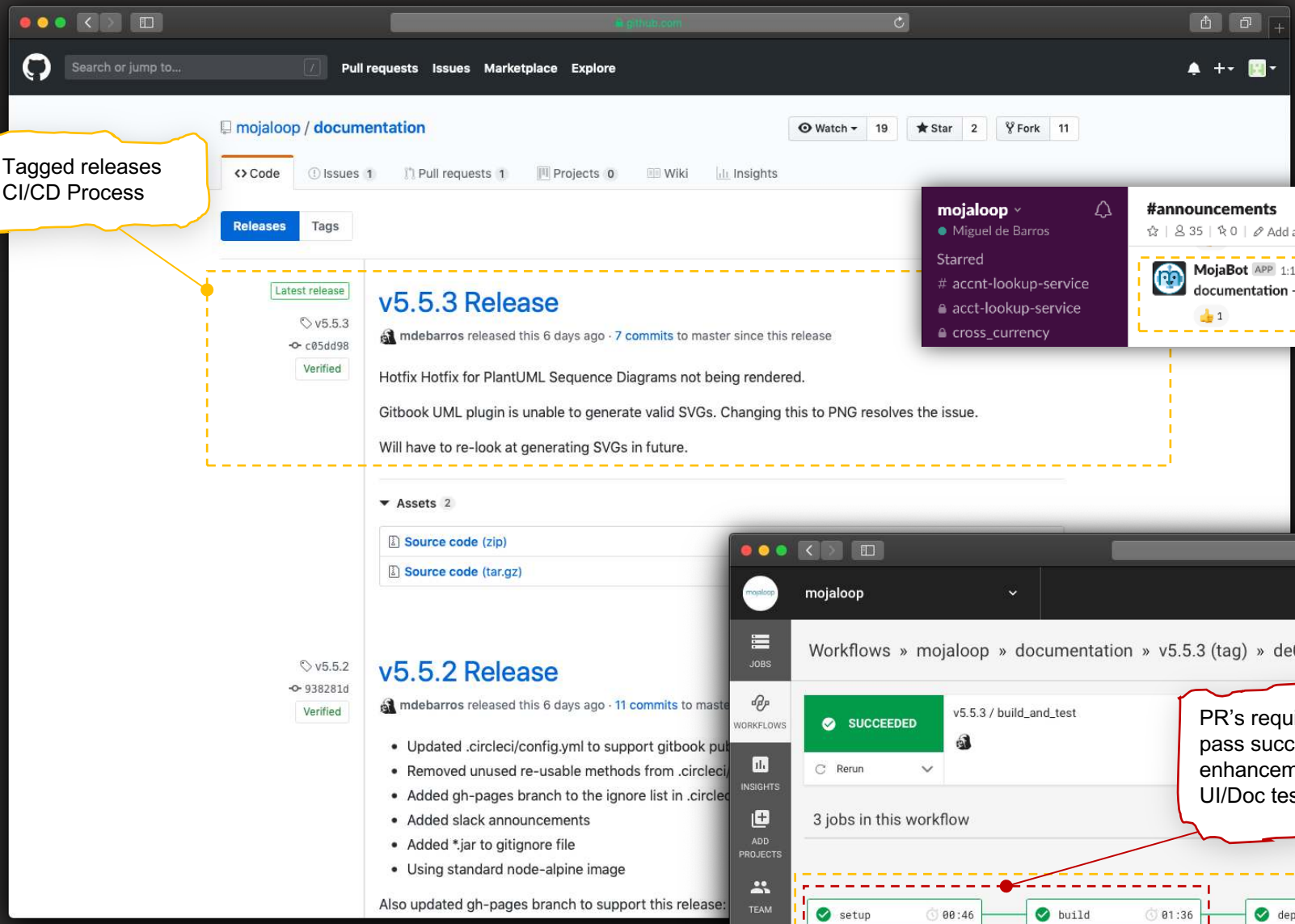
PlantUML are auto-generated as part of build process.

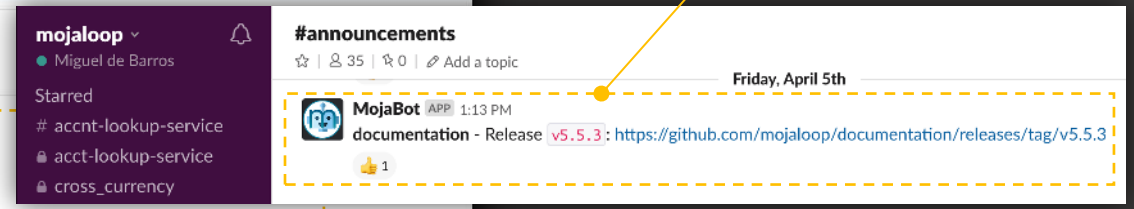


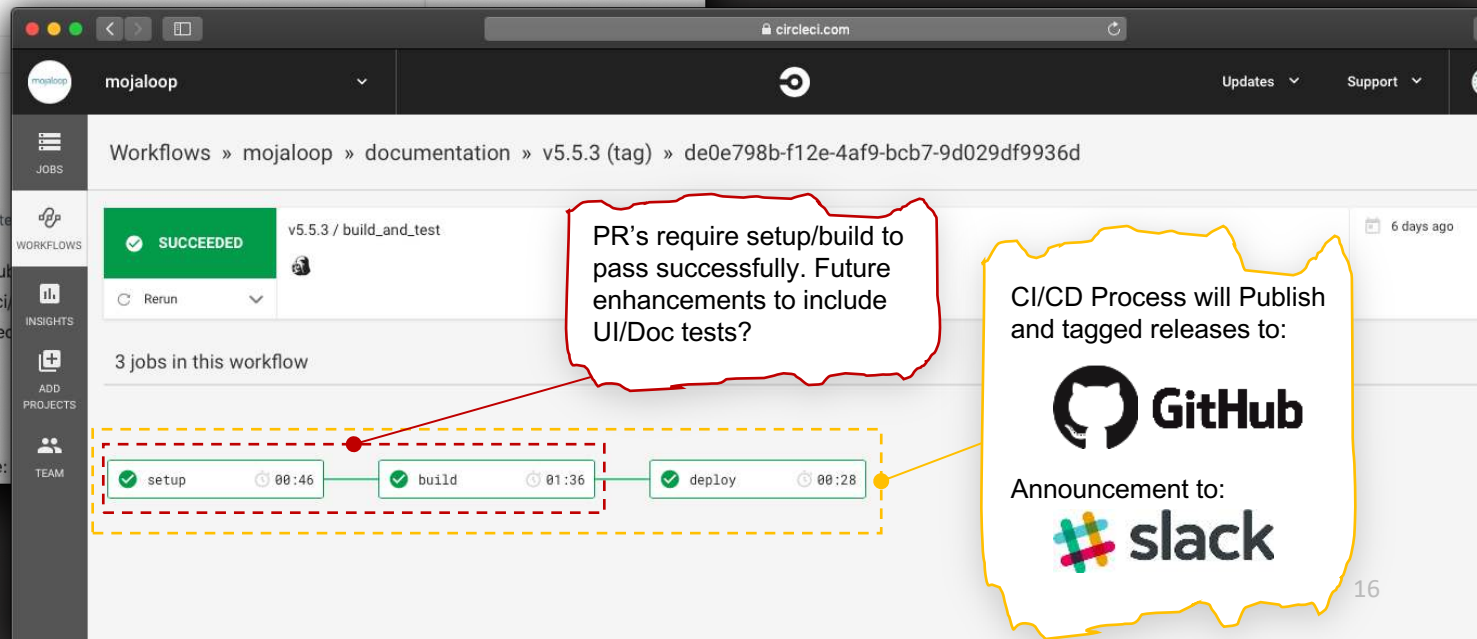
Documentation - CI/CD

PRs & Tagged releases trigger CI/CD Process

slack announcements on successful deployment.







PR's require setup/build to pass successfully. Future enhancements to include UI/Doc tests?

CI/CD Process will Publish and tagged releases to:

GitHub

Announcement to:

slack

Documentation: Wiki (Spanish)

Motivations (started on May 2018):

1. Sparse and deprecated documentation
2. Hard deployment process
3. Confusing concepts (context and translation)
4. New team members onboarding
5. Work logging

Documentation: Wiki (Spanish)

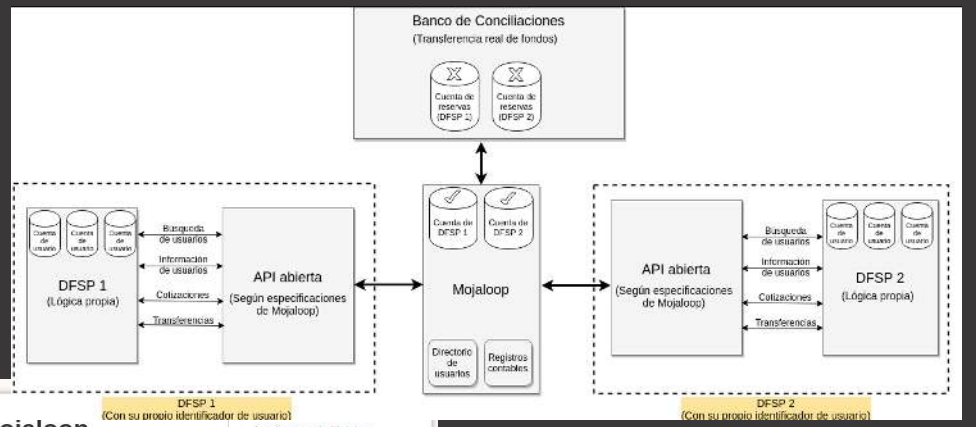
Structure

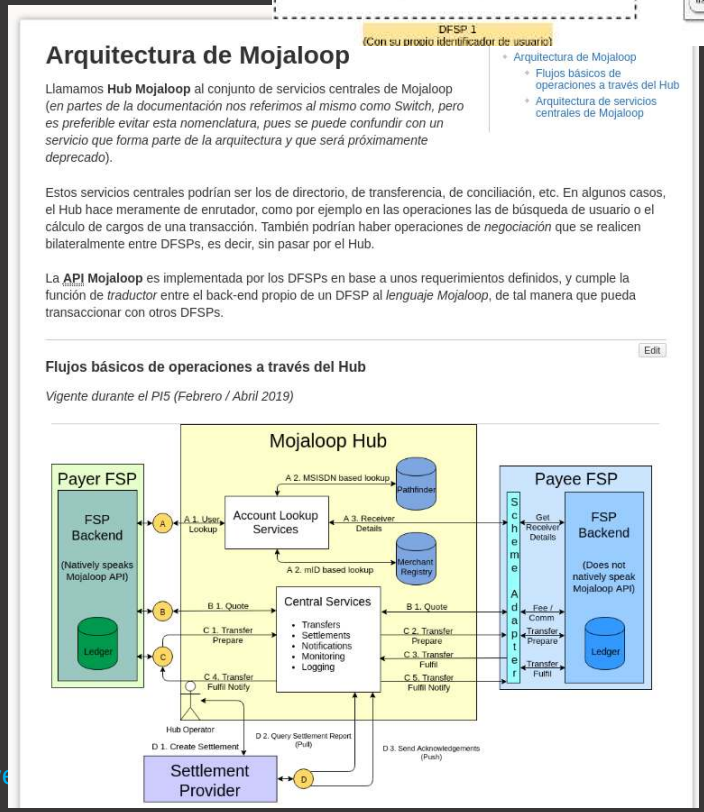
1. Introduction (Onboarding)
2. Infrastructure/DevOps
3. Specific topics
4. Related business topics

Documentation: Wiki (Spanish)

Onboarding

1. Level One Principles
2. Use cases
3. Architecture (main services + Open API)
4. Basic concepts
   a. Hub, scheme, DFSP…
   b. Settlement, position, NDC…
   c. Interledger, quote, fulfilment…





Arquitectura de Mojaloop

Llamamos **Hub Mojaloop** al conjunto de servicios centrales de Mojaloop (*en partes de la documentación nos referimos al mismo como Switch, pero es preferible evitar esta nomenclatura, pues se puede confundir con un servicio que forma parte de la arquitectura y que será próximamente deprecado*).

Estos servicios centrales podrían ser los de directorio, de transferencia, de conciliación, etc. En algunos casos, el Hub hace meramente de enrutador, como por ejemplo en las operaciones las de búsqueda de usuario o el cálculo de cargos de una transacción. También podrían haber operaciones de *negociación* que se realicen bilateralmente entre DFSPs, es decir, sin pasar por el Hub.

La **API Mojaloop** es implementada por los DFSPs en base a unos requerimientos definidos, y cumple la función de *traductor* entre el back-end propio de un DFSP al *lenguaje Mojaloop*, de tal manera que pueda transaccionar con otros DFSPs.

Edit

Flujos básicos de operaciones a través del Hub

Vigente durante el PI5 (Febrero / Abril 2019)

Documentation: Wiki (Spanish)

Infrastructure/DevOps

1. Kubernetes, Rancher, Docker, Helm
2. Environment setup
   a. Single-node (Minikube) vs multiple nodes
   b. Specs (Memory, CPU cores, disk space)
3. Hacks
   a. K8s, Docker, Helm commands...
   b. Monitoring

Edit

Inicios de 2019

A inicios de 2019 se agrega otro servidor sobre un hipervisor VMware. El objetivo del servidor es contar con recursos suficientes para realización de pruebas de conceptos. Las características del servidor son las siguientes:

Procesador	12 cores
HDD	500 GB
RAM	32 GB
OS	CentOS 7
IP	192.168.0.72

— Características de servidor de testing (2019)

Sobre el segundo servidor, también se utiliza virtualización anidada para segmentar los recursos internos disponibles para diferentes objetivos. El hipervisor utilizado para el efecto es QEMU-KVM. El primer servidor que es creado con el fin de disponibilizar una zona DNS para el proyecto y además disponibilizar un repositorio. Este servidor cuenta con:

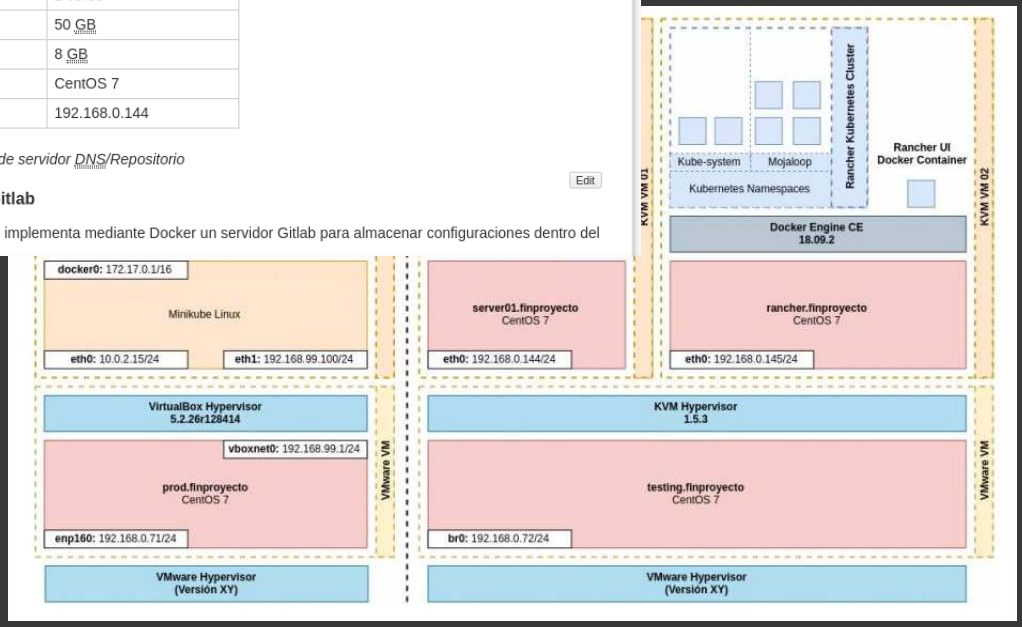
Procesador	2 cores
HDD	50 GB
RAM	8 GB
OS	CentOS 7
IP	192.168.0.144

— Características de servidor DNS/Repositorio

Edit

El repositorio Gitlab

Posteriormente, se implementa mediante Docker un servidor Gitlab para almacenar configuraciones dentro del servidor



Documentation: Wiki (Spanish)

Specific topics

1. Open API
    a. Lookup, quote, transfer, settlement
    b. Basic P2P flow + postman collection
2. Settlements
    a. Theory + models (Glenbrook research)
3. Cross-border
    a. Proposal
4. Security

Edit

Servicios y recursos (endpoints) de la API

Recurso	Servicio	Descripción
/participants	Búsqueda de un participante	Determinar en qué FSP está ubicada la contraparte de una transacción financiera.
/parties	Obtener información sobre un participante	Obtener información sobre la contraparte de una transacción financiera.
/quotes	Cotizar una transacción	Calcular todas las partes de una transacción que modificarán el monto de la transacción (cargos y comisiones). Este recurso se utiliza para cotizar una única transacción, es decir, de un Emisor a un Receptor.
/transfers	Realizar una transferencia	Ejecutar propiamente una transacción financiera transfiriendo los fondos electrónicos del Emisor al Receptor, posiblemente a través de sistemas intermediarios. Este recurso se utiliza para realizar una única transacción, es decir, de un Emisor a un Receptor.
/transactionRequests	Solicitar una transacción	El Receptor solicita una transferencia al Emisor. El Emisor puede aprobar o rechazar esta transacción. Una aprobación de la petición, iniciará la transacción financiera propiamente.
authorizations	Verificar autorización	Solicitar al Emisor las credenciales necesarias cuando el mismo inició la transacción desde algún POS, ATM u otro dispositivo en el sistema del FSP Receptor.
transactions	Consultar transacción	Obtener información relacionada a la transacción financiera; por ejemplo, un token creado en una transacción exitosa.
bulkQuotes	Cotizar una transacción	Análoga al recurso */quotes*, pero utilizada para realizar la operación en volumen, es decir, de un Emisor a varios Receptores.
bulkTransfers	Realizar una transferencia	Análoga al recurso */transfers*, pero utilizada para realizar la operación en volumen, es decir, de un Emisor a varios Receptores.

Edit

Documentation: Spanish Wiki (Spanish)

Team effort

1. Challenges faced were shared on the PI4 meeting
2. Documentation team (discussion on structure, topics, roles)
3. Ongoing English translation

Results from the OSS Team effort

1. Deployment guide (PI4)
2. Gitbook (PI5)

Available for the community: https://wiki.fintechinversiones.com.py

Where is Mojaloop being Deployed?

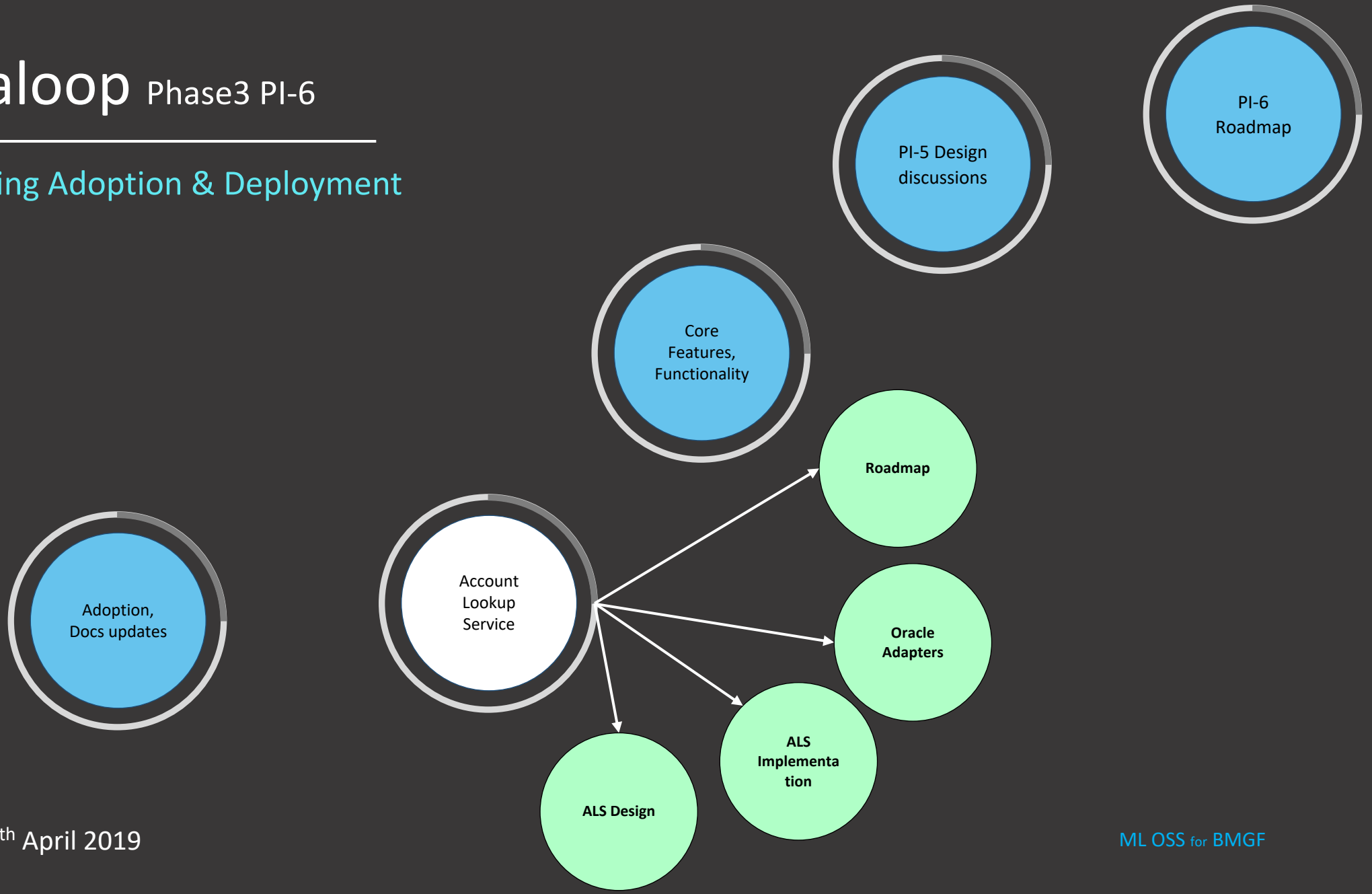
- **Tanzania**: A project sponsored by the Central Bank. This is a merchant and P2P payments solution between banks and mobile money operators, enabling a massive growth in payments beyond what's been achieved on bilateral system integrations.
- **Mowali** : A pan-African payment solution involving Orange and MTN but also anticipating other players.
- **Others** that are confidential to those companies.

PI-5 Account Lookup Service

Supporting Adoption & Deployment

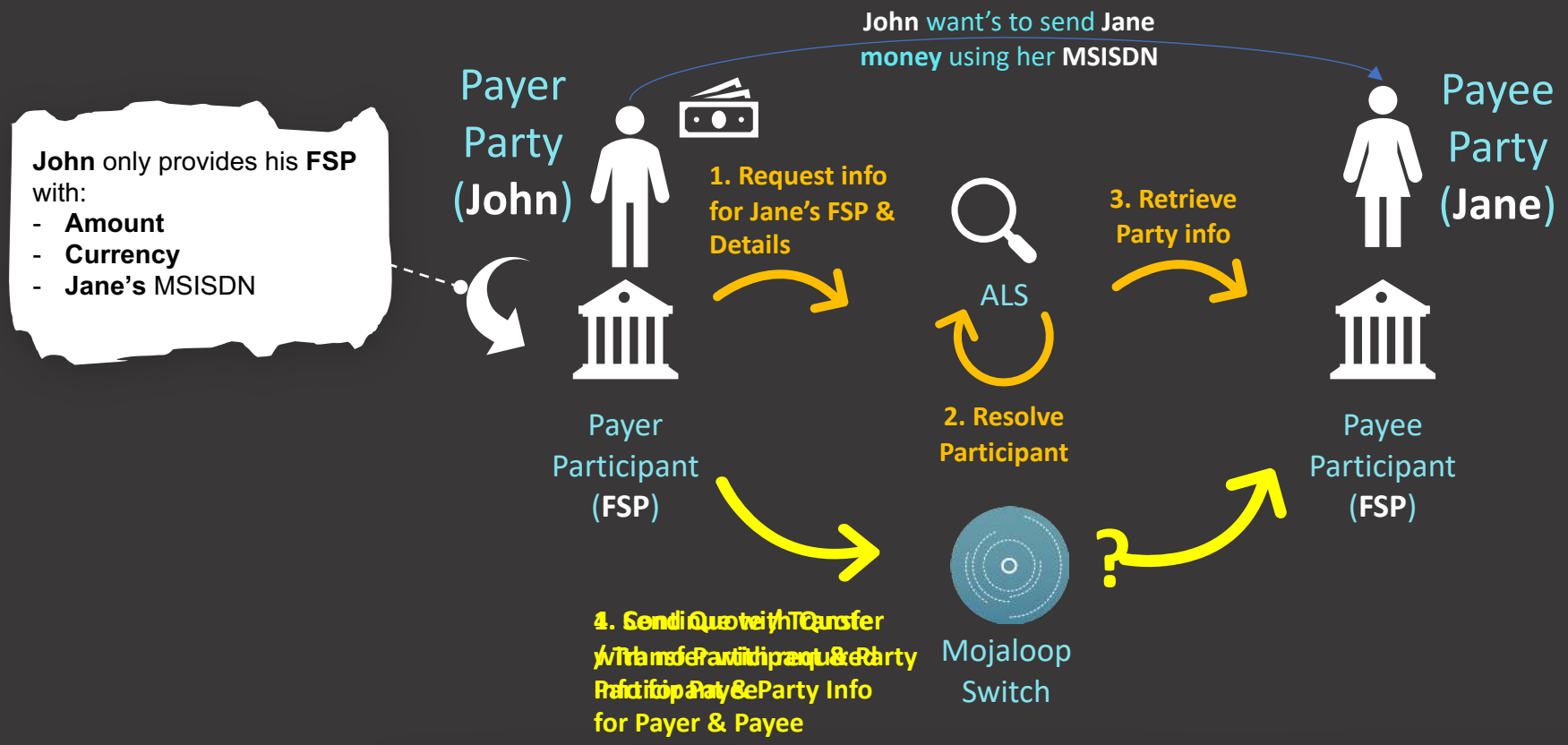
Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



16th – 18th April 2019

ALS: Why do we need the ALS (Account Lookup Service)?



API Definition

Open API for FSP Interoperability Specification

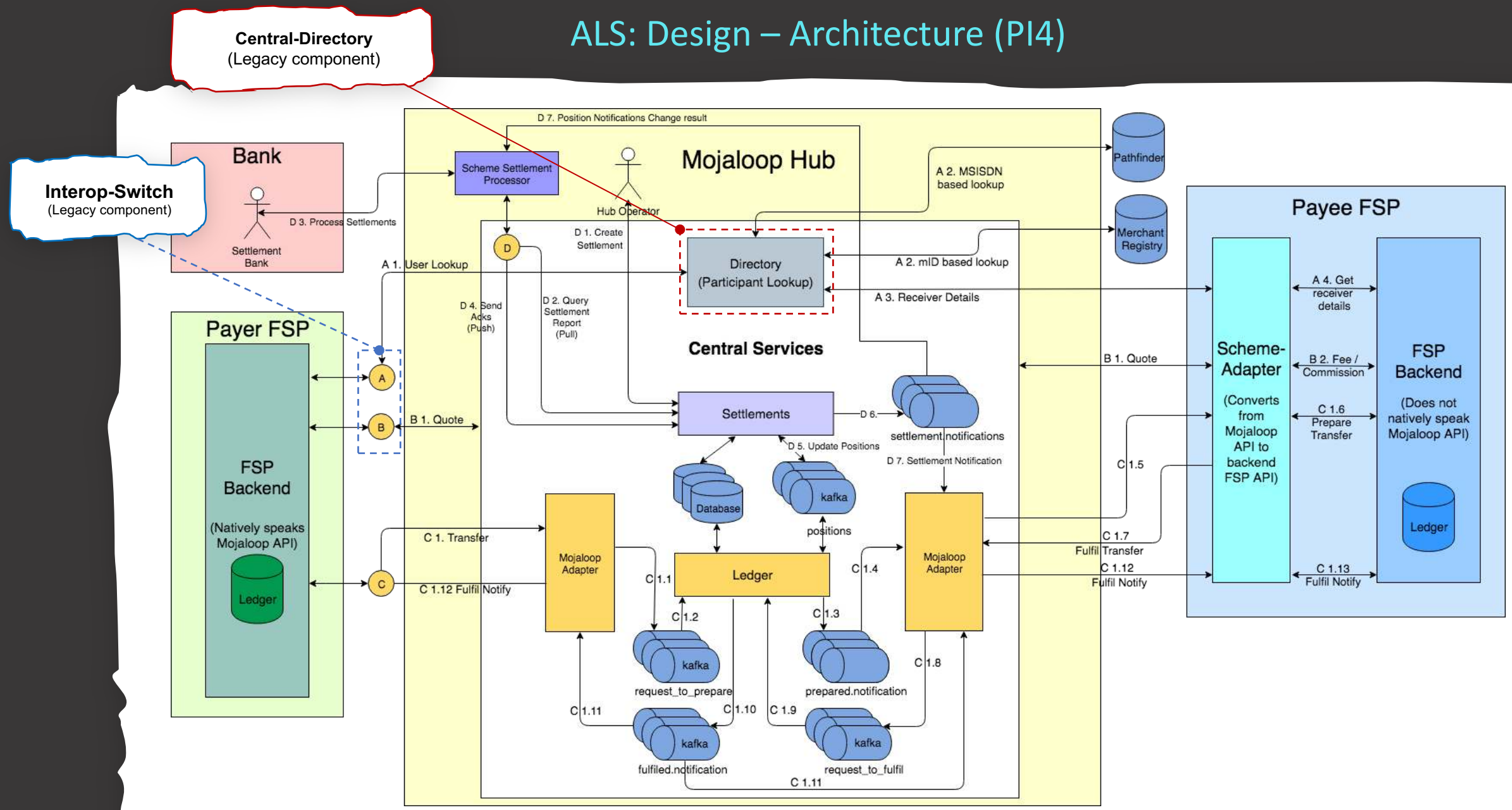
6.1 High Level API Services

On a high level, the API can be used to perform the following actions:

- **Lookup Participant Information** – Find out in which FSP the counterparty in a financial transaction is located.
  - Use the services provided by the API resource **/participants**.
- **Lookup Party Information** – Get information about the counterparty in a financial transaction.
  - Use the services provided by the API resource **/parties**.

Page 76 - https://github.com/mojaloop/mojaloop-specification/blob/master/API%20Definition%20v1.0.pdf

ALS: Design – Architecture (PI4)



ALS: Design – Architecture (Current PI5)

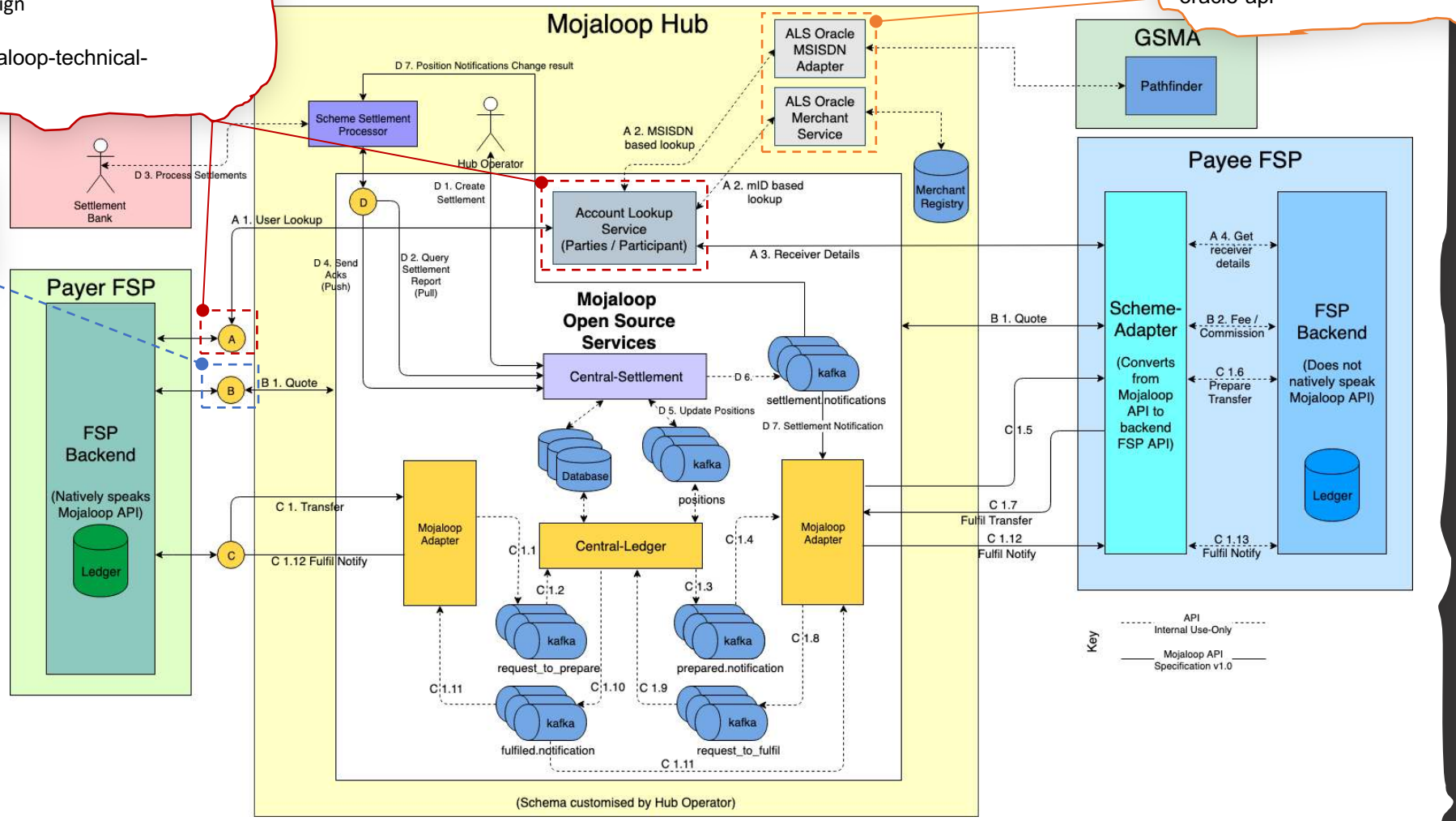
- Deprecated **Central-Directory**
- Deprecated **Interop-Switch** for **Participants**/**Parties**
- **Account Lookup Service (ALS)** design and implementation
- ALS code-base **contributed** from **Mowali.**
- **Re-factored** OSS standards and design

http://mojaloop.io/documentation/mojaloop-technical-overview/account-lookup-service/

Legacy **Interop-Switch** continues to handle **Quote** operations

Oracle Adapter/Service:
- Standard interface contract (openapi/swagger)

http://mojaloop.io/documentation/api/#als-oracle-api



ALS: Design - Overview

Key
[●] Fully implemented
[●] Legacy Code
[●] Partially implemented
[●] Not implemented
[○] Out of Scope

ALS Mojaloop API operations

Parties

[●] GET - Request
[●] PUT - Response
[●] PUT - Error
[●] SubId support

Participants

[●] GET - Request
[●] POST - Create
[●] POST - Create (Bulk)
[●] PUT - Response
[●] PUT - Error
[●] DEL - Delete
[●] SubId support

ALS Admin API operations

Oracles

[●] POST – Create
[●] GET - Request
[●] PUT - Update
[●] Delete - Remove

Design Considerations

Account Lookup Service (ALS)

- Pluggable Architecture to support multiple Participant Registries
- Scalable Architecture
- Future considerations:
  - Support running a single ALS with multiple Switches
  - Leverage on routing mechanisms defined by Cross-Border/Network design for Switch routing
- Implements Participants operations for Mojaloop API specifications v1.0
- Implements Parties operations for Mojaloop API specifications v1.0

Oracle Service/Adapter

- Standard interface/contract based on Mojaloop API specifications v1.0
- Support resolution of FSPIOP-Destination for FSPs via the ALS
- Synchronous API that follows traditional REST Paradigms
- Accelerator (Template) to expedite development
- Community Contributions by TIPS Team:
  - ALS Oracle Template - https://github.com/mojaloop/als-oracle-template

Oracle API operations

Participants

[○] GET - Request
[○] POST - Create
[○] POST – Create (Blk)
[○] PUT - Update
[○] Delete - Remove

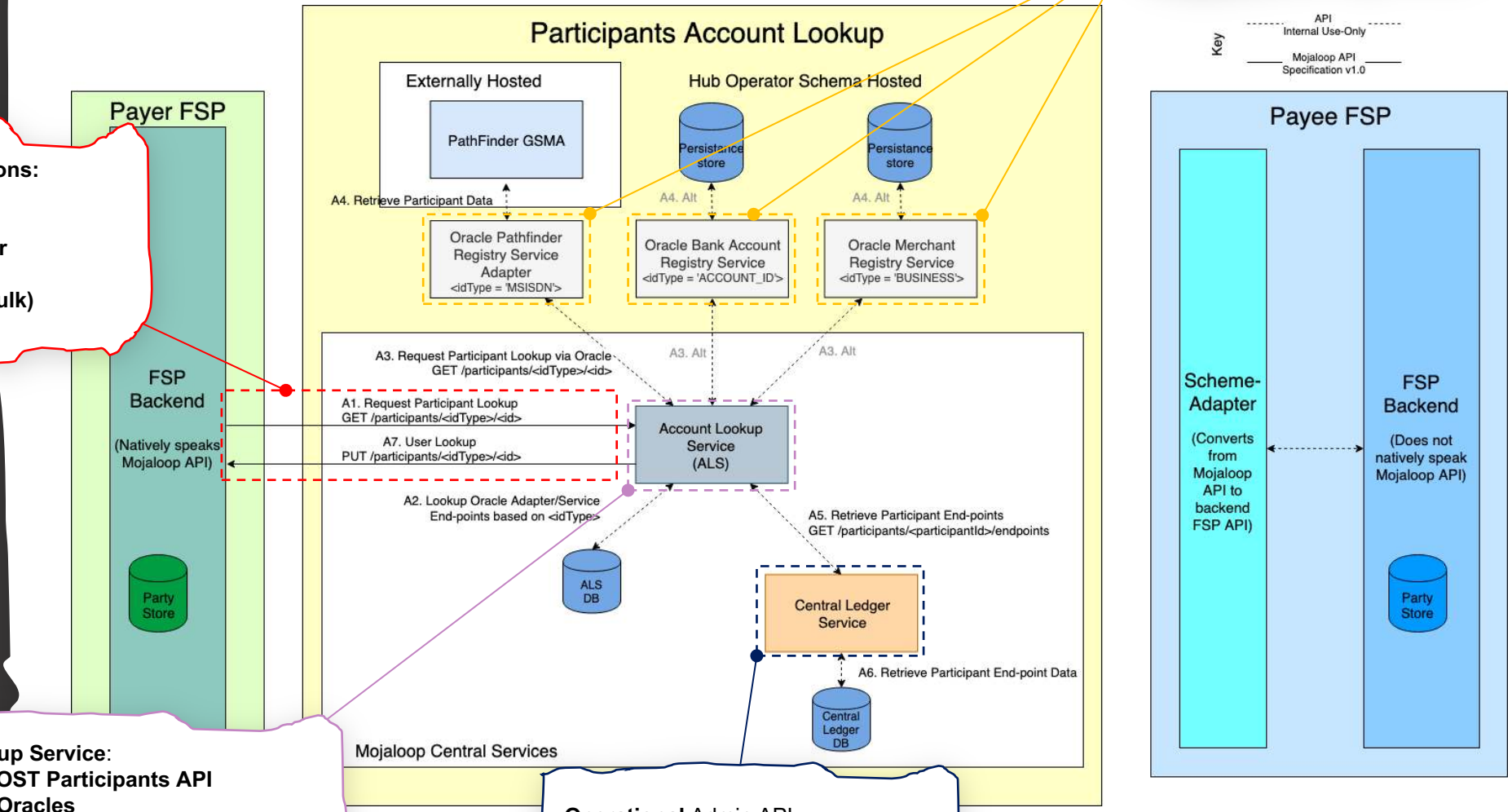
● Interim:
- no validations applied to “update” operations
- full design for Participants POST – Create is pending

* http://mojaloop.io/documentation/mojaloop-technical-overview/account-lookup-service/

ALS: Design – Participants

Oracle Adapter/Service:
- **Registries to resolve Participant requests** via an easy pluggable sync API.

API Mojaloop Specifications:
- **GET /Participants**
- **PUT /Participants**
- **PUT /Participants/error**
- **POST /Participants**
- **POST /Participants (Bulk)**
- **DEL /Participants**



Account Lookup Service:
- **GET/PUT/POST Participants API**
- **Routing to Oracles**
- **Callback resolution for FSP** responses

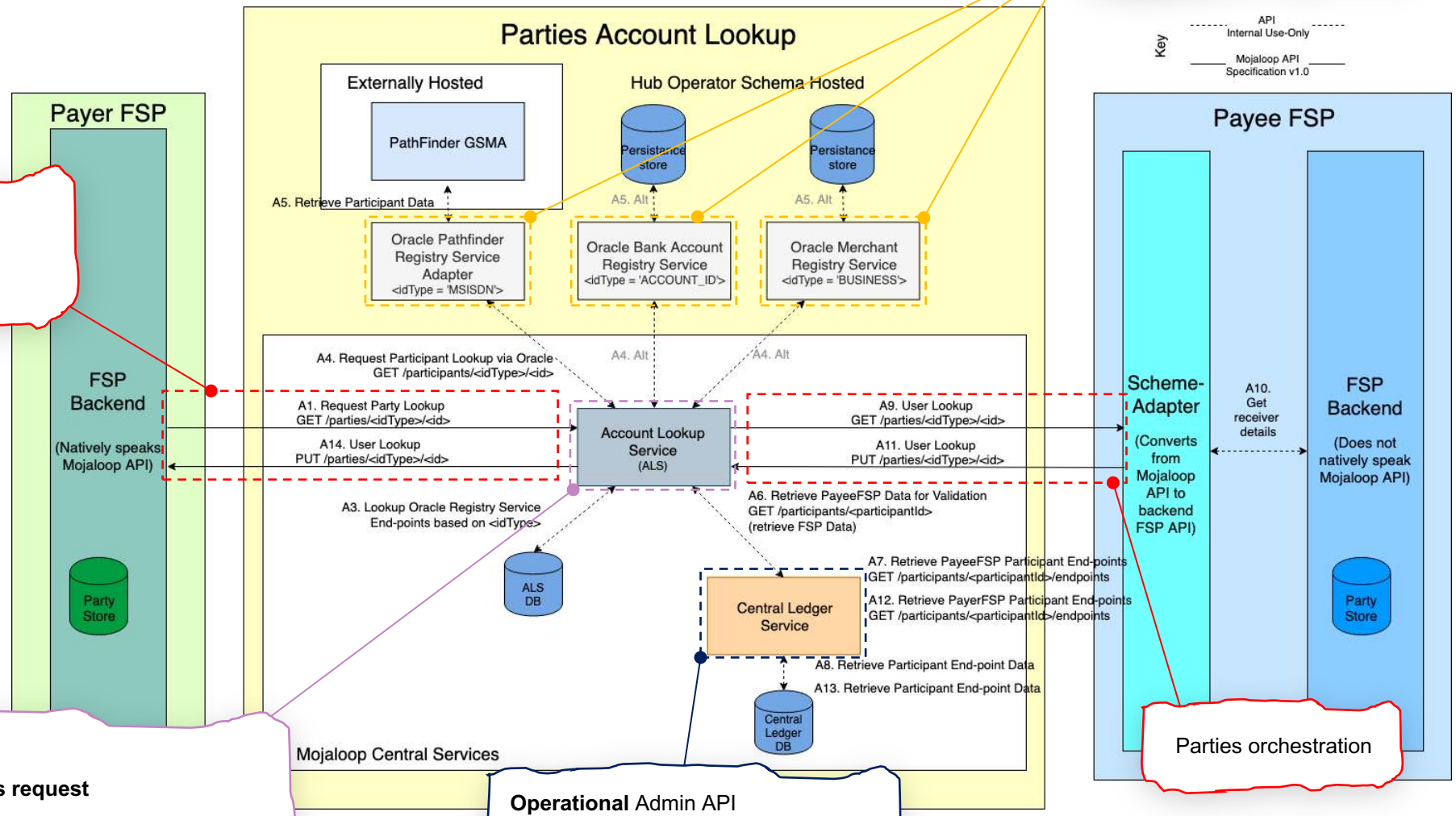
Operational Admin API
- **resolve FSP Callback End-points**

* http://mojaloop.io/documentation/mojaloop-technical-overview/account-lookup-service/

ALS: Design – Parties

Oracle Adapter/Service:
- **Registries to resolve Participant requests** via an easy pluggable sync API.

API Mojaloop Specifications:
- **GET /Parties**
- **PUT /Parties**
- **PUT /Parties /error**



Account Lookup Service:
- **GET/PUT Parties**
- **Orchestration of the Parties request**
- **routing to Oracles**
- **Callback resolution for FSP** responses.

Operational Admin API
- **resolve FSP Request End-points**
- **resolve FSP Callback End-points**

Parties orchestration

* http://mojaloop.io/documentation/mojaloop-technical-overview/account-lookup-service/

ALS: Roadmap

Account Lookup Service (ALS)

- Functional
  - Support for SubId on Participants operations
  - Support for SubId on Parties operations
  - Support to run ALS as a Central service with multiple Switches
    - Leverage routing mechanisms defined by Cross-Border/Network design for Switch routing
- Non-Functional
  - Expand on Unit Test coverage
  - Integration tests
  - QA Framework
  - Performance Tests

Oracle Service/Adapter

- Registry for MSISDN via Pathfinder (UDP) leveraging existing Mock-Pathfinder implementation
- Enhance Oracle Template Accelerator to provide functionality to auto-generate a base project

Simulator

- Enhance to provide Participants simulation
- Enhance to provide Parties simulation

ALS: Bank Account Oracle

Bank Account Oracle (BAO)

- Functional
  - Identify Bank Participant
  - Uses standardized account numbers
  - Account Number format (14 digits)
    - 3 digits: Participant Identifier
    - 1 digit: Currency Identifier
    - 10 digits: Party Account Identifier

  - Participant registry is maintained (based on the 3 digits participant identifier)
  - Support for one currency (TZS)

Operations Supported

Participants

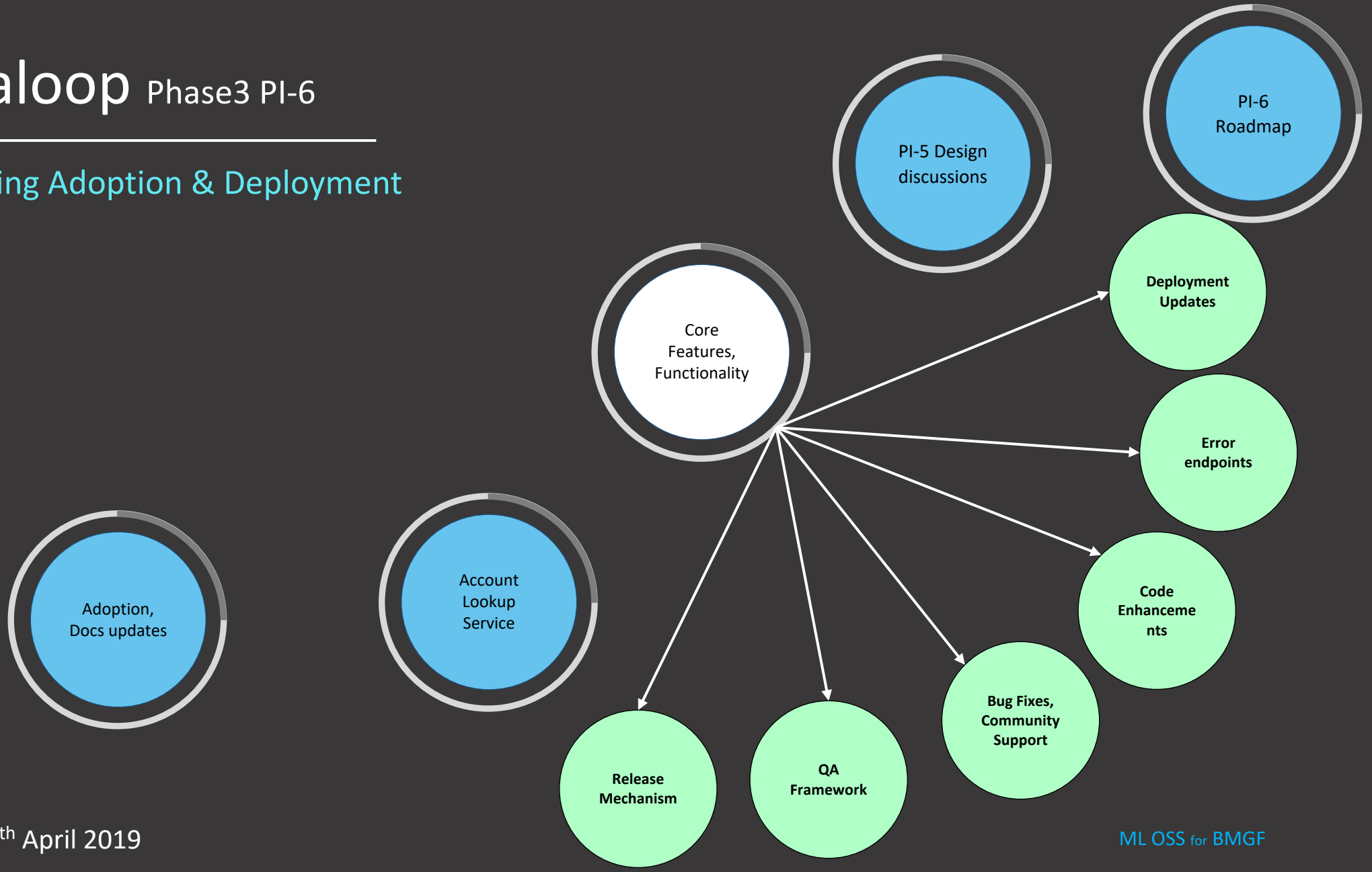
[●] GET - Request
[●] POST - Create
[●] POST - Create (Bulk)
[●] PUT - Update
[●] DELETE - Delete

ALS: Implementation

Demo

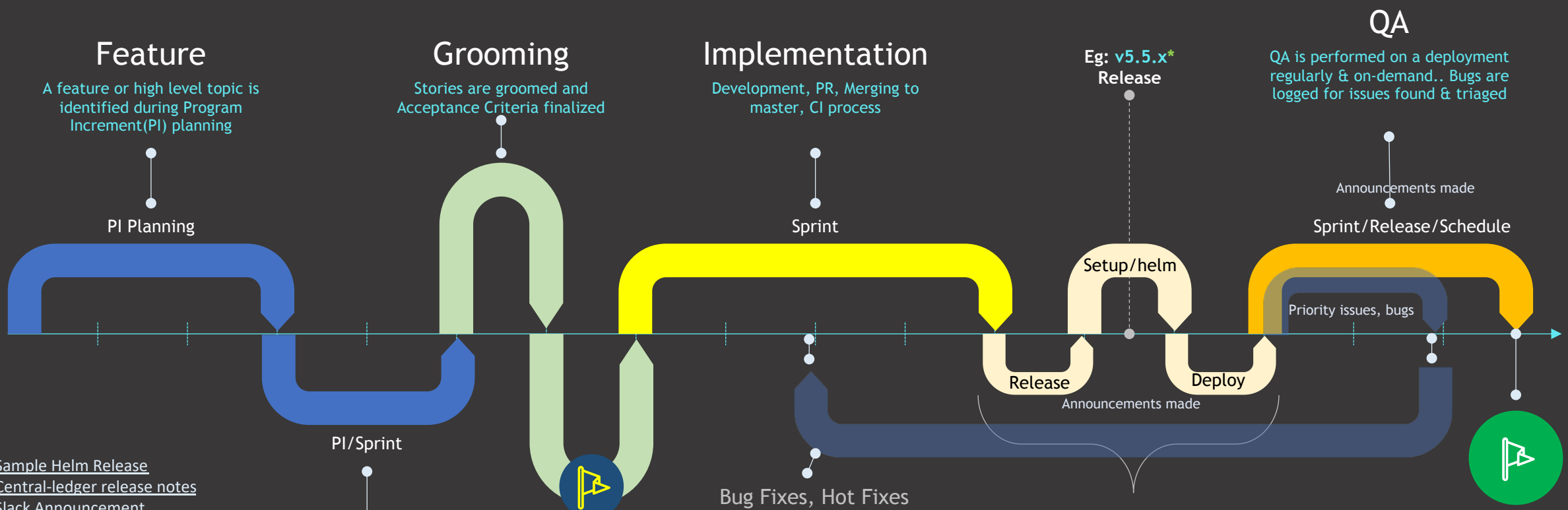
Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



16th – 18th April 2019

ML OSS: Release Mechanism



- Sample Helm Release
- Central-ledger release notes
- Slack Announcement

* Inspired by SemVer; x.y.z – where 'x' is the PI number, 'x.y' is the Sprint number and 'z' is the number of release in a particular sprint

Note: This depicts a single iteration that happens multiple times over the course of a PI

QA Framework: Regression Testing

is not

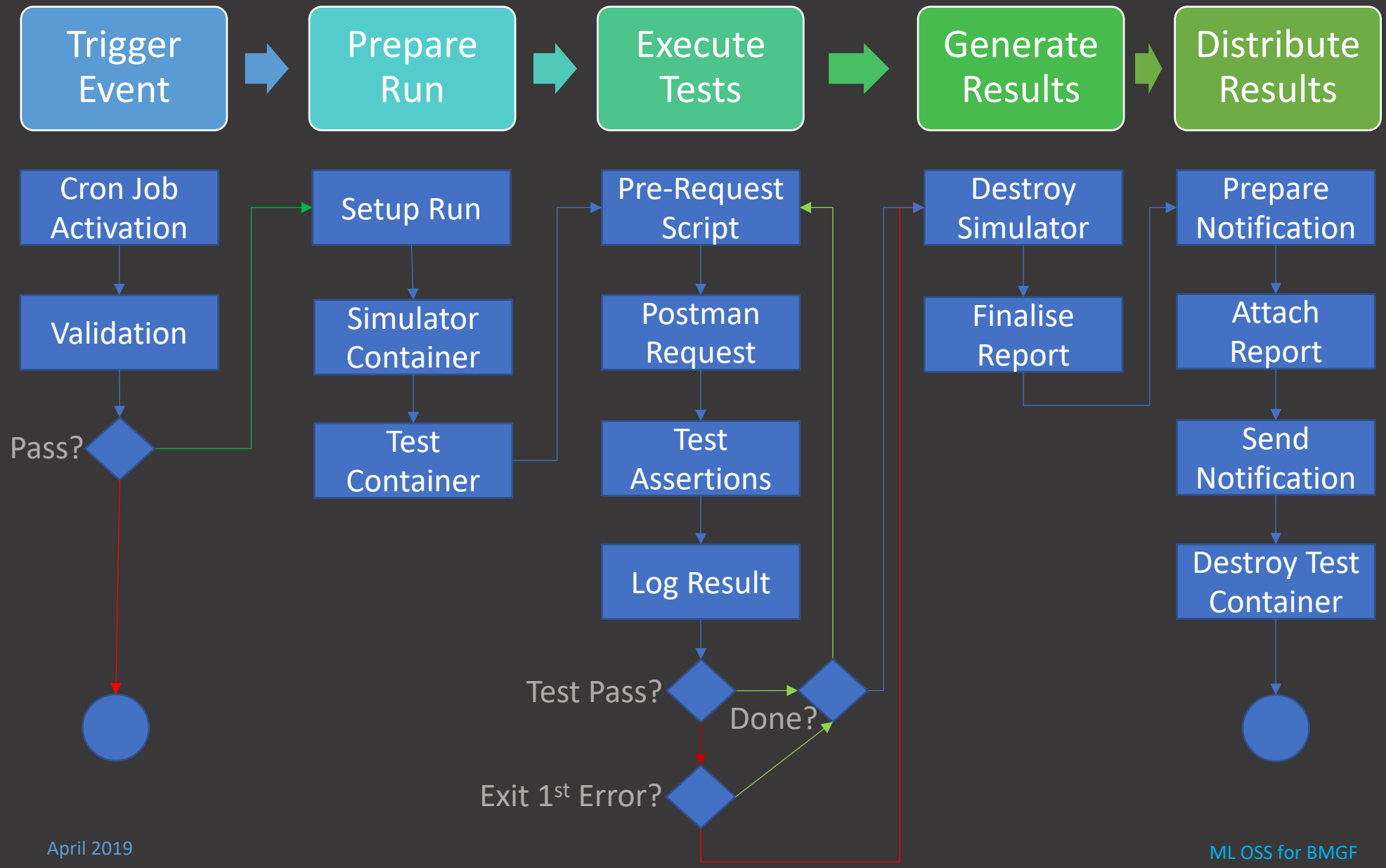
1) Unit testing
2) Coverage Testing
3) Integration Testing
4) Specific Functionality (use case) testing

QA Framework: Regression Testing

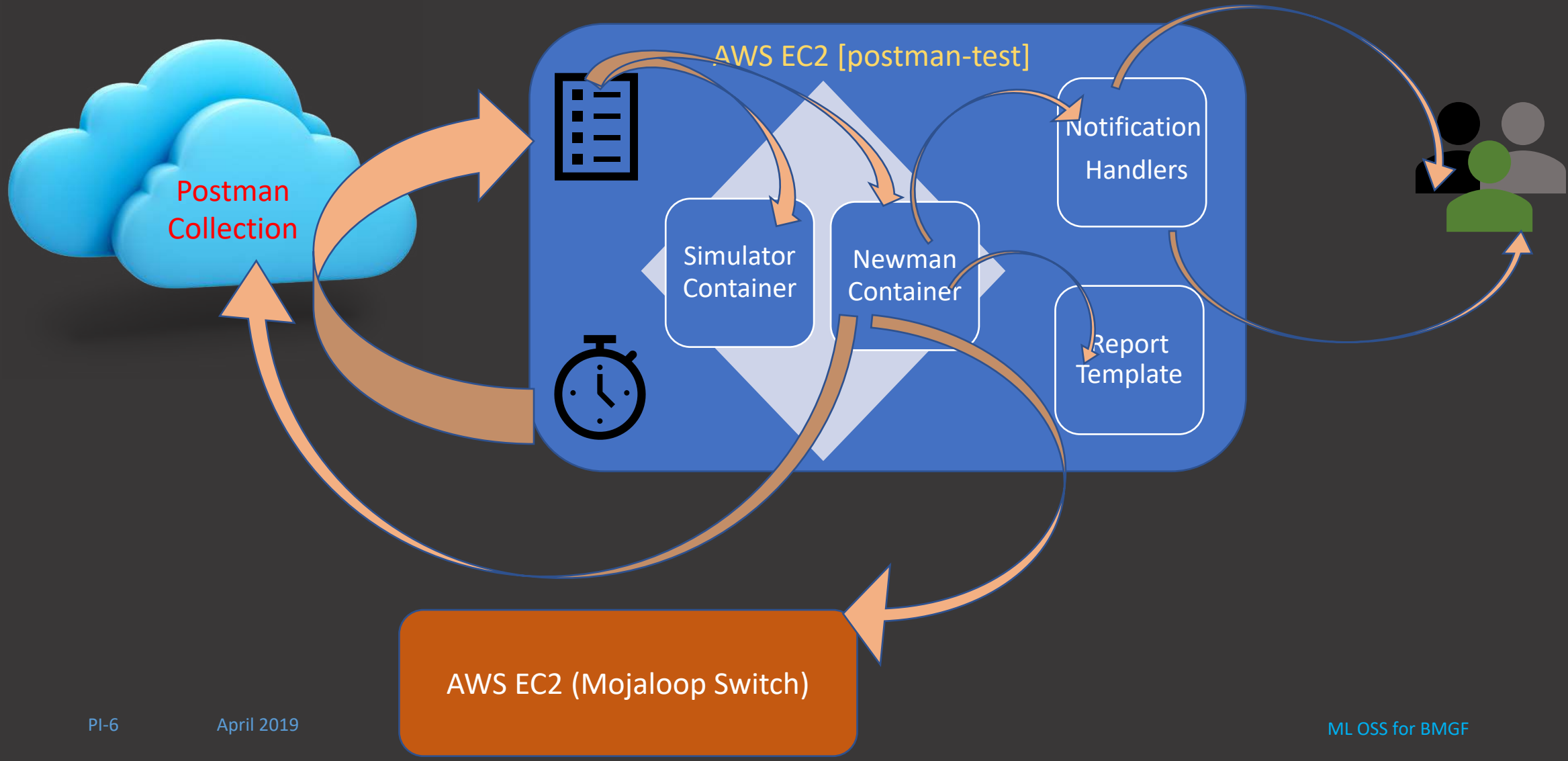
is typically characterized by

1) Ensuring a specified deployed environment performs and produces consistent results
2) Manual or automatic triggering by linking it with an event
3) Execute as many times as required – idempotent
4) Automatically sends out alerts in case of any failure or unexpected result
5) Communicated results can be used to pinpoint anomalies
6) Executes the entire range of functionality of the system under test (FR / NFR)

Regression Testing – Instance Life-Cycle



Regression Testing Interaction



Bug Fixes, Community Support

QA

~40 Bugs Logged

QA: Mowali, ML Core Team, Community

Bugs Triaged

14 High

Bugs are triaged. 14 high priority bugs in PI5

Fixed

Bugs fixes

Bugs are added to Sprints and fixed based on priority

Deployed

Released, Deployed

After fixed are made, releases are made and deployed to AWS Dev

Closed

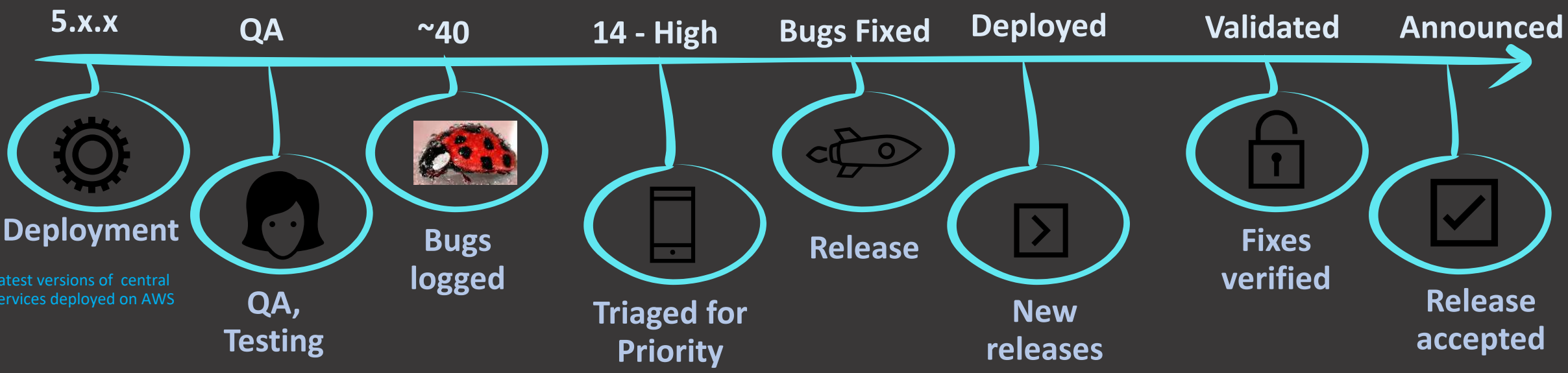
Validated, closed

Fixes for bugs are validated on a deployment and closed if fixed

ZERO

High Priority Bugs

The latest release **5.5.2** has zero high priority bugs open



Note: This depicts a single iteration that happens multiple times over the course of a Sprint/PI

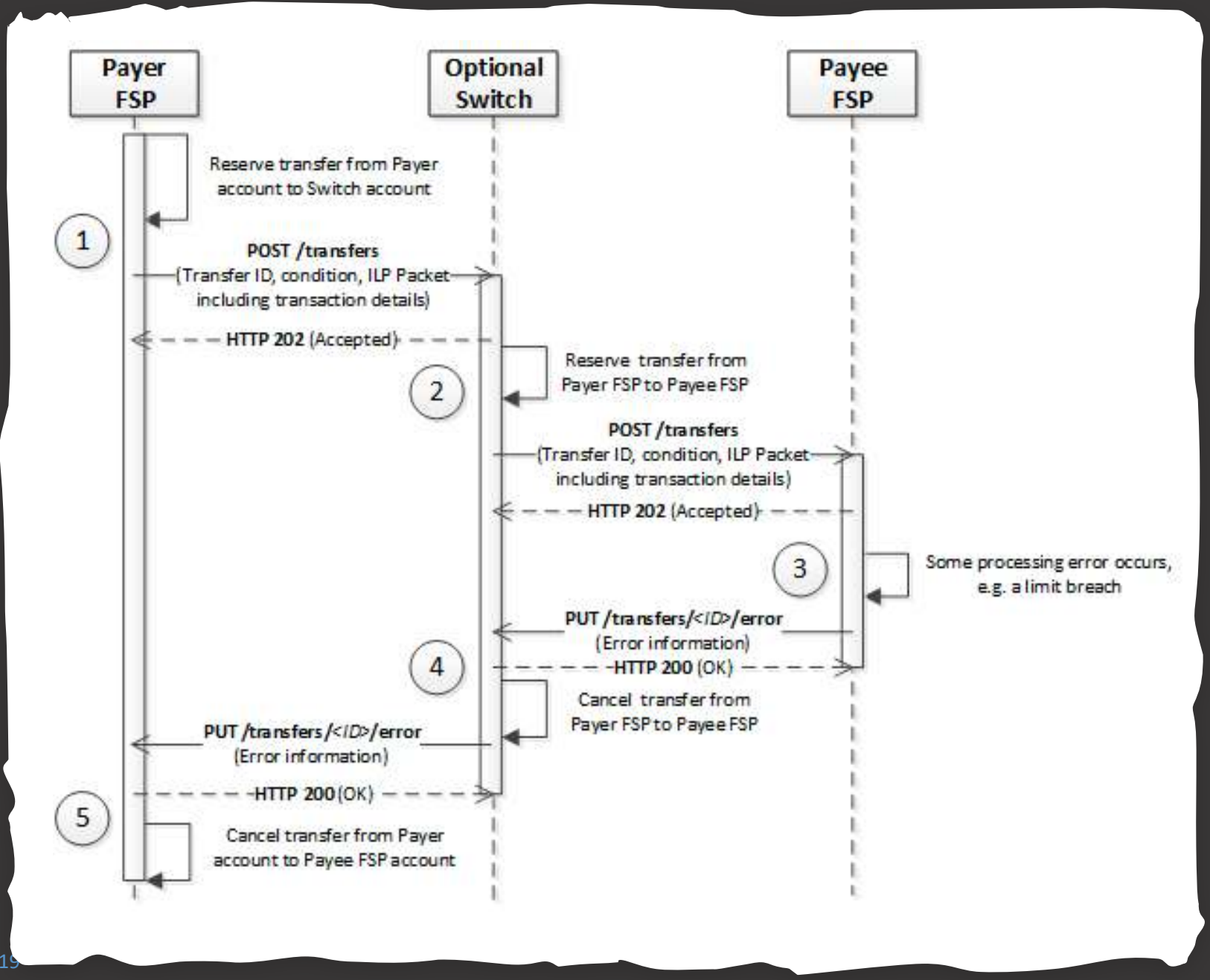
Community Support

1. Slack *#general* channel
2. Support to current adopters, implementers & contributors [Bug Fixes, Priority Features]
3. Documentation updates
4. Addressing Specification questions
5. Addressing Deployment questions
6. Adding of FAQs Section

Error end-points on the Switch

1. Generic handling of error callbacks on /transfers
2. Scenarios
    1. Payee sending PUT /error callback for a transfer
    2. Using ’ABORTED’ as transferState in a fulfilment
3. Key design decisions
    1. Clarify usage of ‘ABORTED’ state by Payee in a Fulfil request

PI-5: Handling of error callback for transfers



Integration tests

1. PI-4 Central-Ledger integration test processing time: > 10m
    1. The wait for callback result implemented via a While – Timeout loop
2. PI-5 Central-Ledger integration test processing time: < 5m
    1. Implemented `async-retry` library to handle the wait for callback result which allows for:
        1. Retry count limit
        2. Min timeout
        3. Max timeout

Future enhancements:

- Apply similar methodology to Integration Tests on ML-API-Adapter

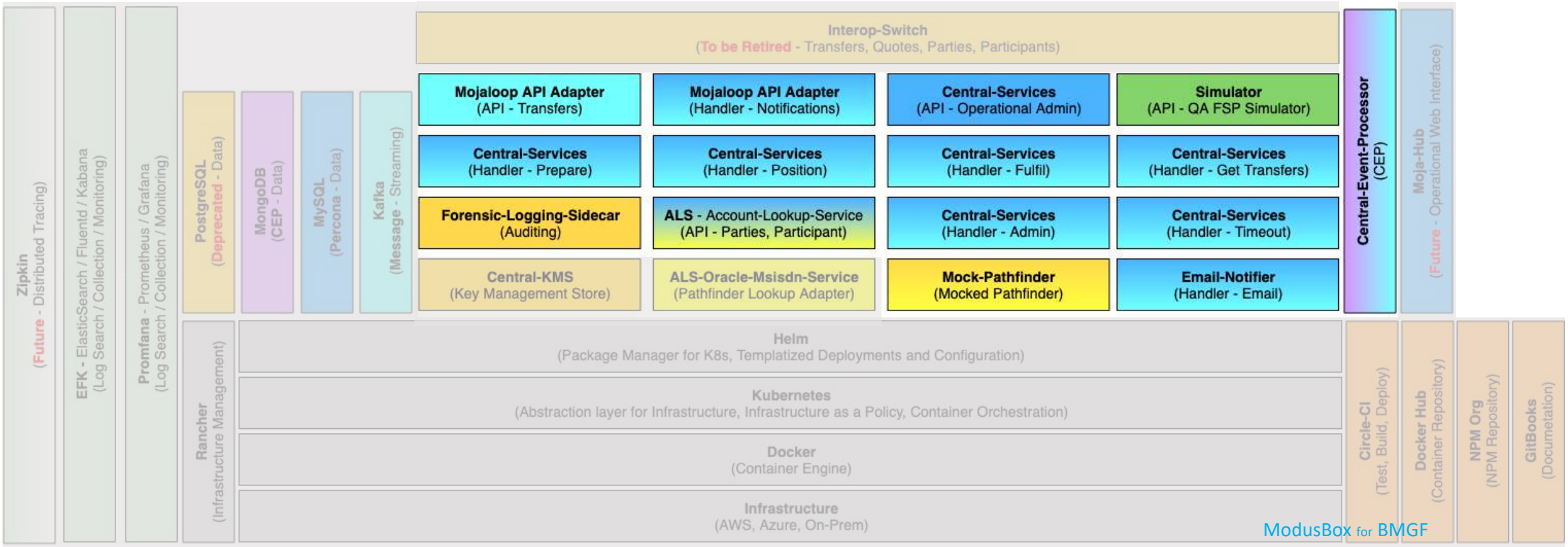
Validation on incoming requests

1. FSPIOP-Source validation
    1. Validate source FSP exists and is valid
    2. Validate Source FSP is a participant of an existing Transfers (GET – Request / PUT - Callback)
    3. Validate Source FPS is the Party owner for Participant Delete operations
2. FSPIOP-Destination
    1. Validate Destination FSP exists and is valid
    2. Validate Destination FSP is a Participant of an the Transfers (POST - Create/ PUT - Callback)

Node upgrade

1. All Node Central Components have been upgraded from Node v8.x to the latest LTS v10.x
    1. Security *fixes
    2. Performance *improvements (http, event-loop)

* https://nodejs.org/en/blog/release
* https://medium.com/sharenowtech/node-js-10-is-the-lts-the-enterprise-will-love-2395372a80c3



ModusBox for BMGF

Metrics enhancements

1. Standardized metric naming conventions enforced through-out code-base on central-ledger and ml-api-adapters: moja_<process>_<sub-process>
2. Labels were added to identify the specific processing component
3. Example metrics:
    1. moja_transfer_prepare (labels: serviceName="central-handler-prepare" or "ml_service")
    2. moja_transfer_position (labels: serviceName="central-handler-position")
    3. moja_transfer_fulfil (labels: serviceName="central-handler-fulfil" or "ml_service")
    4. moja_notification_event (labels: serviceName="ml-handler-notification")

Improved Code Standardization, Quality & Maintenance

1. Hard-coded values were re-factored to use enumerations for ml-api-adapter & central-ledger (future story to create an enum framework across micro-service components to fix duplication of enums across components).
2. Improvements on routing and call-back handling between the ml-api-adapter and central- ledger:
    1. Standard Header transformer for ml-api (specific to mojaloop transport):
        1. Remove unwanted headers (e.g. FSPIOP-Signature when FSPIOP-Source=”switch”)
        2. Set correct headers depending on scenario (e.g. FSPIOP-Method: POST, PUT, etc for the correct use-case prepare, fulfil, etc)
    2. Destination/Source are set by the central-ledger (except for the additional call-back scenario for fulfils which is handled by the ml-api - in future this will be optional config)

Deployment Updates

1. Externalized configuration files into a template file to improve maintenance
2. Cleaned up all configuration files by removing duplicate/unused values
3. Upgraded dependent charts for:
    1. Kafka (upgraded from v1.0.1 to v2.1.0 – improvements: memory, resilience, security, DNS)
    2. Zookeeper (fixed scaling bugs with deployments)
    3. Percona-Xtra-db Mysql (fixed issues with metrics, nodeSelector & tolerations)
4. Added new Helm scripts for :
    1. Documentation
    2. Account-lookup-service
5. Setup a new DEV1 environment behind a VPC (requiring VPN access):
    1. operational management plane (Rancher)
        1. Now Highly Available
        2. Scalable to support future clusters/lab environments
        3. Certificates are auto-provisioned & auto-renewed by cert-manager

PI-5 Objectives Review

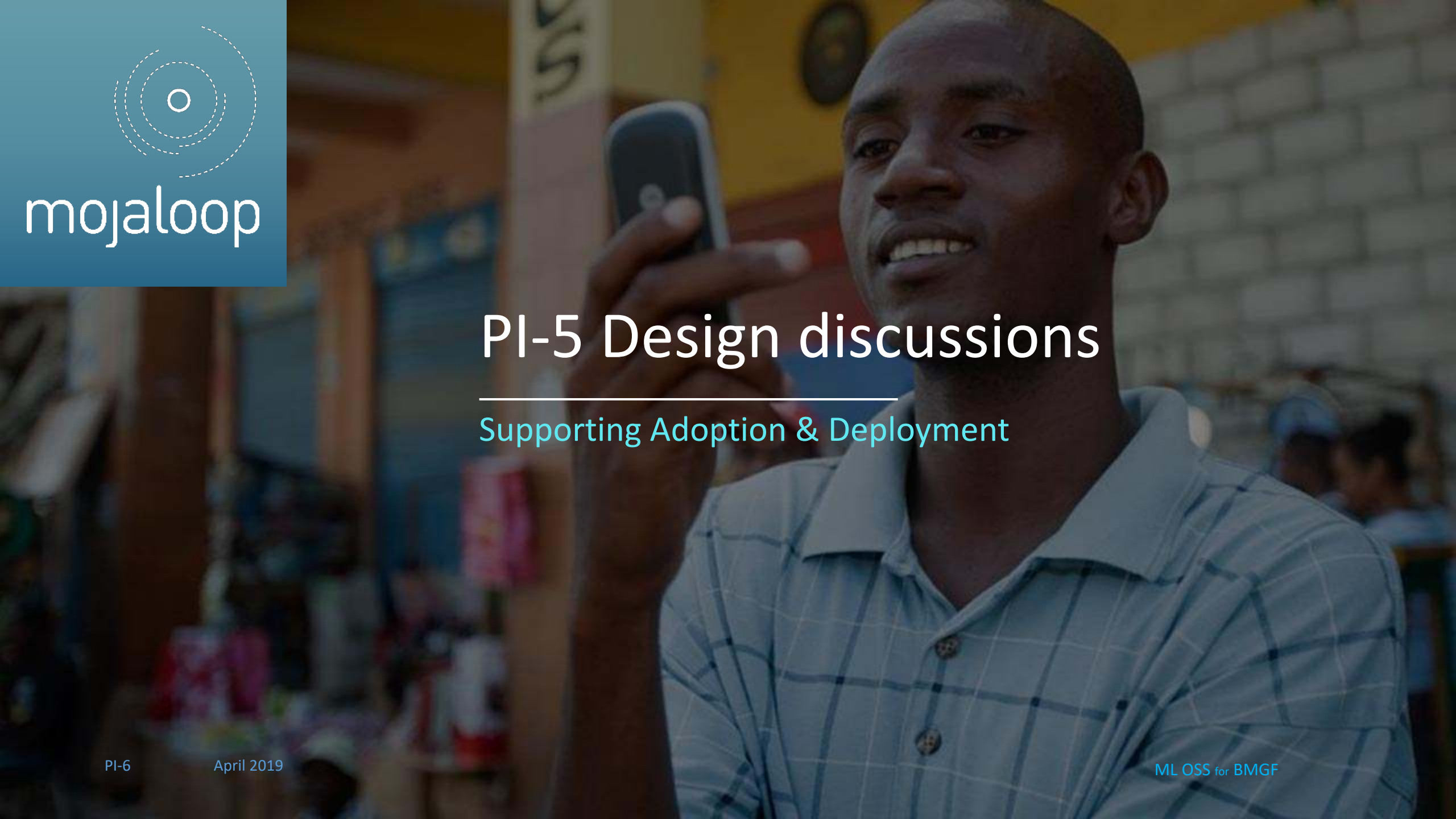
Supporting Adoption & Deployment

PI-5 Objectives Review - 1

1. Account look-up service is adopted and integrated
2. DFSP Handler provisioning is designed, a PoC implemented
3. Standardized Settlements API, Framework
4. Error handling: Supporting error end-points for transfers
5. Providing a standardized simulator that conforms to the API Spec 1.0 (Mowali)
6. Solution for Bulk Payments is designed (to be implemented in PI-6)
7. Support for Community requests regarding Specification, Deployment and adoption
8. *WSO2 API Gateway is supported for transfers messages*
9. Optimistic Goal: Support for Merchant ‘Request to Pay’

PI-5 Objectives Review - 2

1. ALS Overhaul: ALS is adopted, integrated and made comprehensive
2. DFSP Handler provisioning
3. Standardized Settlements API – Background work done
4. Error handling: Supporting error end-points for transfers
5. Node Upgrade, *Code Enhancements*
6. Solution for Bulk Payments is designed (to be implemented in PI-6)
7. Support for Community requests regarding Specification, Deployment and adoption
8. *Documentation, QA enhancements*
9. *Bug Fixes, Reliability, Streamlining Release process*

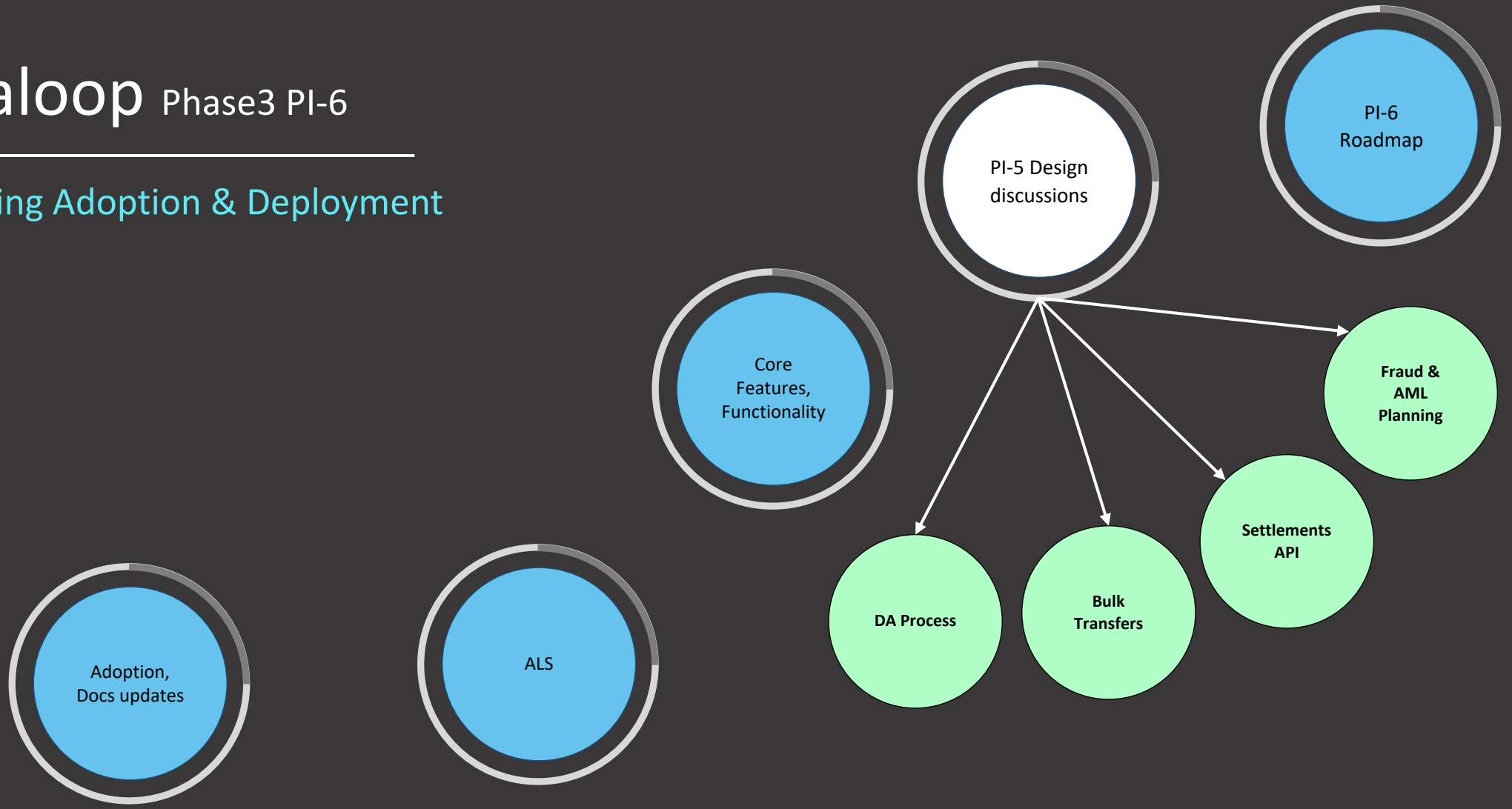


PI-5 Design discussions

Supporting Adoption & Deployment

Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



16th – 18th April 2019

Design Authority

Aims to

1) Ensure a uniform Architecture
2) Create a channel for bringing ideas to Design and Verification
3) Define Technical Strategies
4) Verify Architectural Standards
5) Drive Design Methodologies

Design Authority

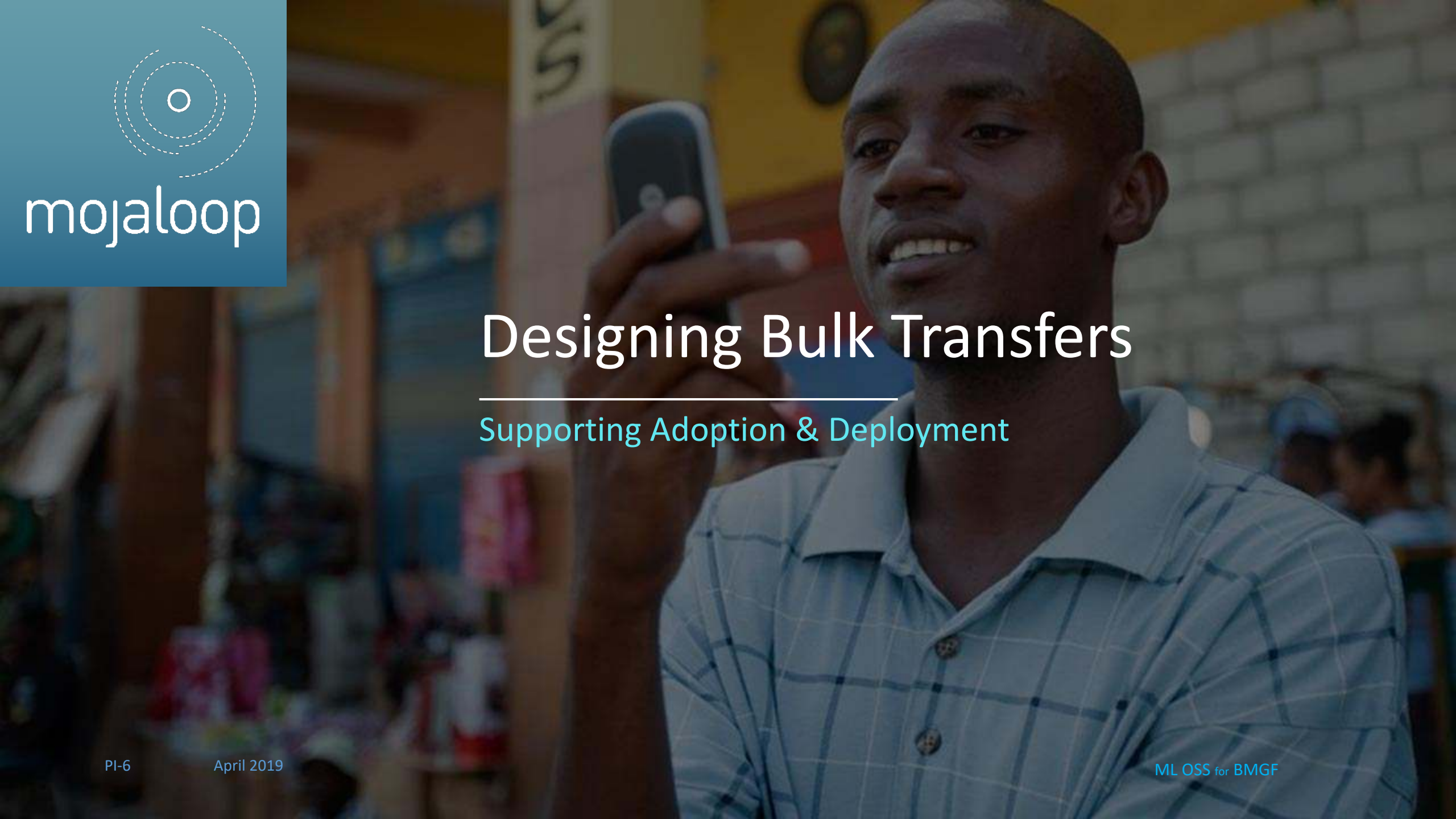
Implementation

1) Have representation from OSS Core Implementer
2) Include representatives from commercial implementers and adopters (similar to scrum-of-scrums)
3) Should ideally have a DA/Design team for each Implementation Project
4) Regular scheduled DA meetings as well as Ad-Hoc high priority design decisions
5) Ideally meet in zoom meetings but in rare cases, verification and voting via email or Slack
6) Publicly visible Decision Log with clear life-cycle management of decisions
7) Invite Public and Community participation via easy submission

Proposal for initial membership

Representation

1) BMGF
2) ML Core OSS team
3) Crosslake Technologies
4) Coil
5) Mowali
6) BoT TIPS
7) CCB



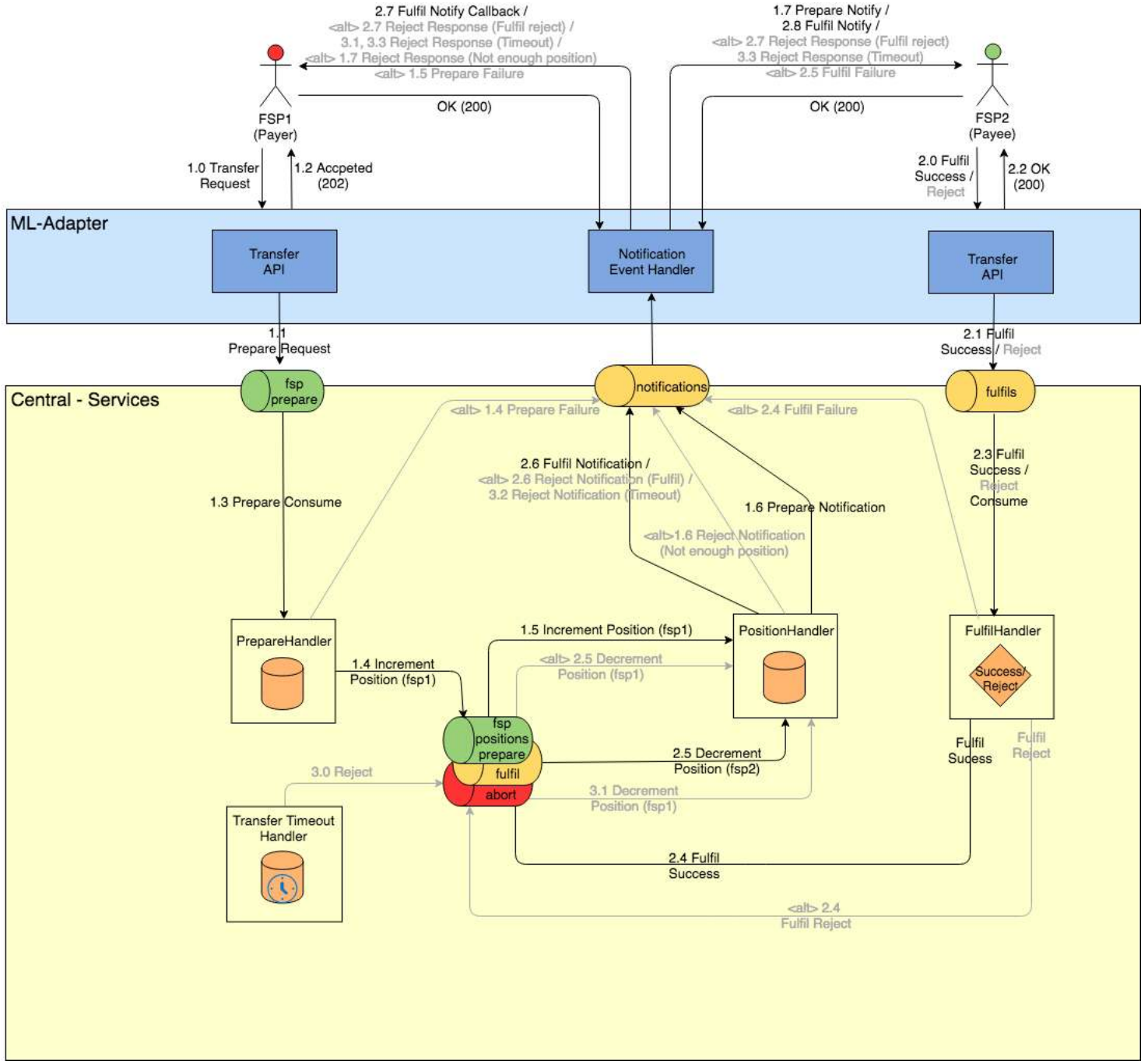
Designing Bulk Transfers

Supporting Adoption & Deployment

PI-5: Bulk Transfers Design

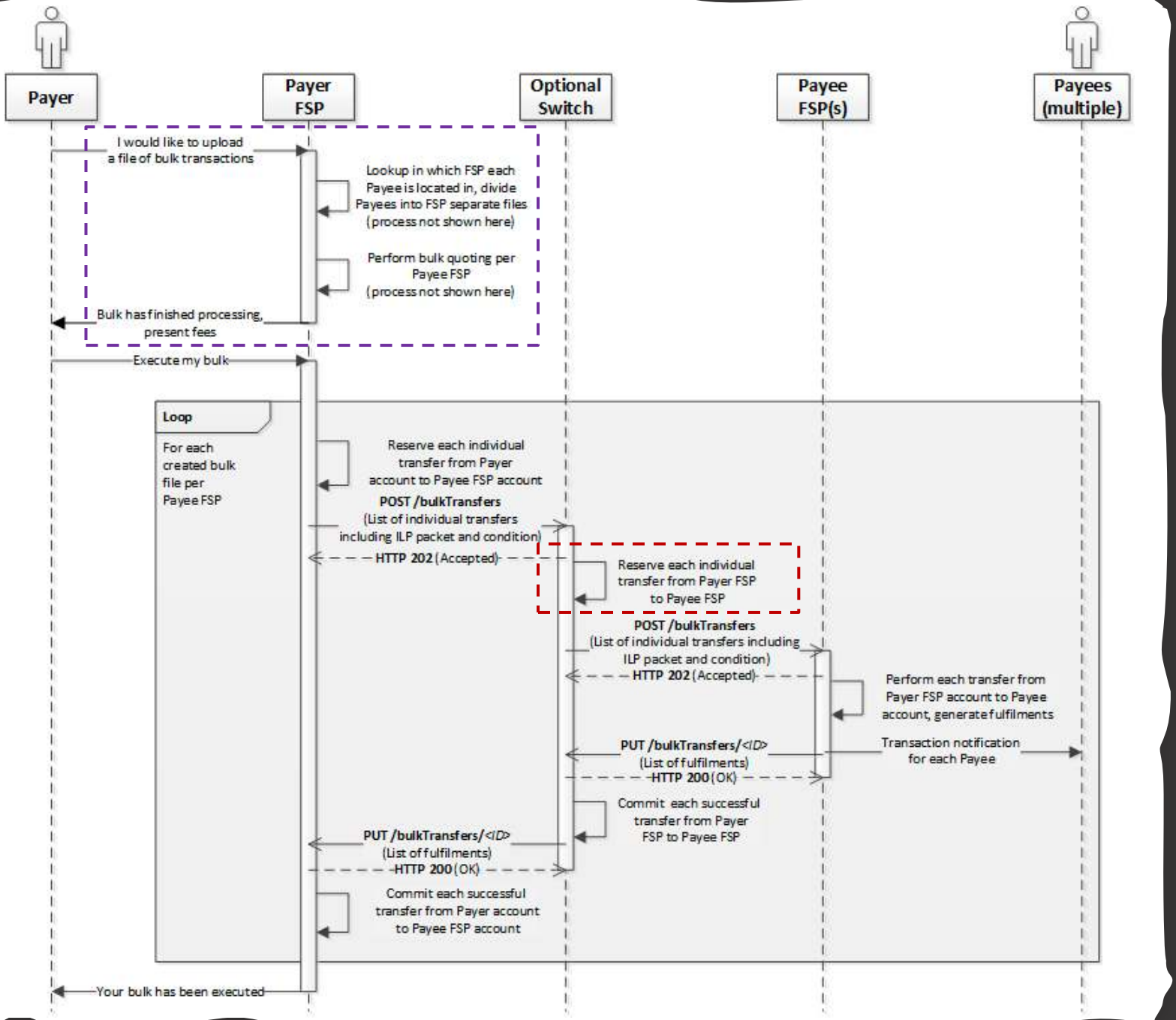
1. Review Individual transfers (Example P2P)
2. Discuss Bulk Transfers Sequence from the Specification
3. Considerations during Design
4. High Level architecture diagram
5. Items to consider for implementation
6. Roadmap

PI-5: Individual Transfers High Level Architecture

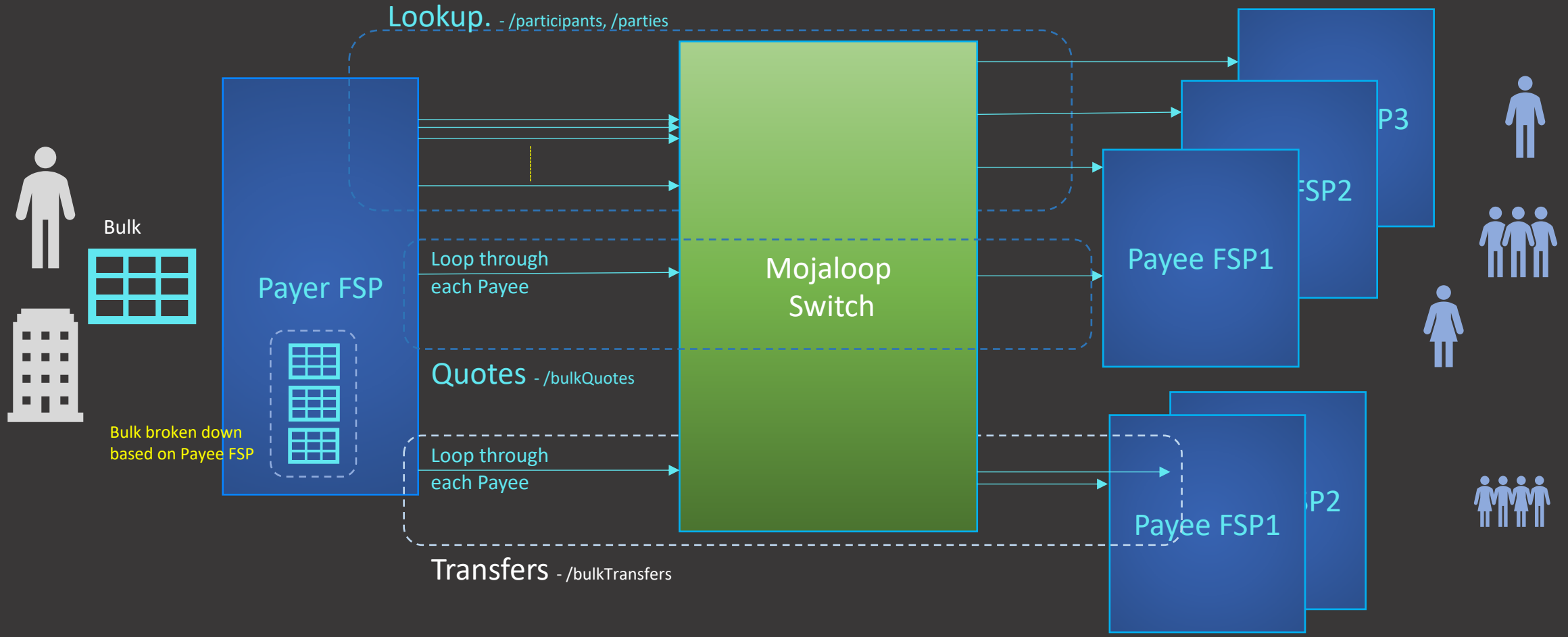


PI-6

PI-5: Bulk Transfers Sequence from the Spec



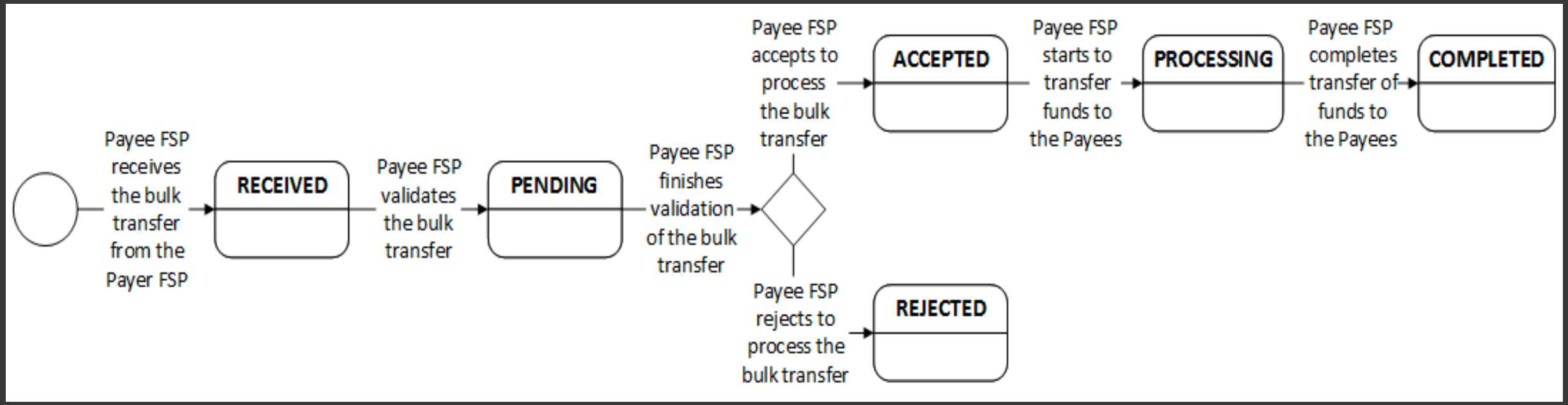
PI-5 Bulk Transfers: Break-down



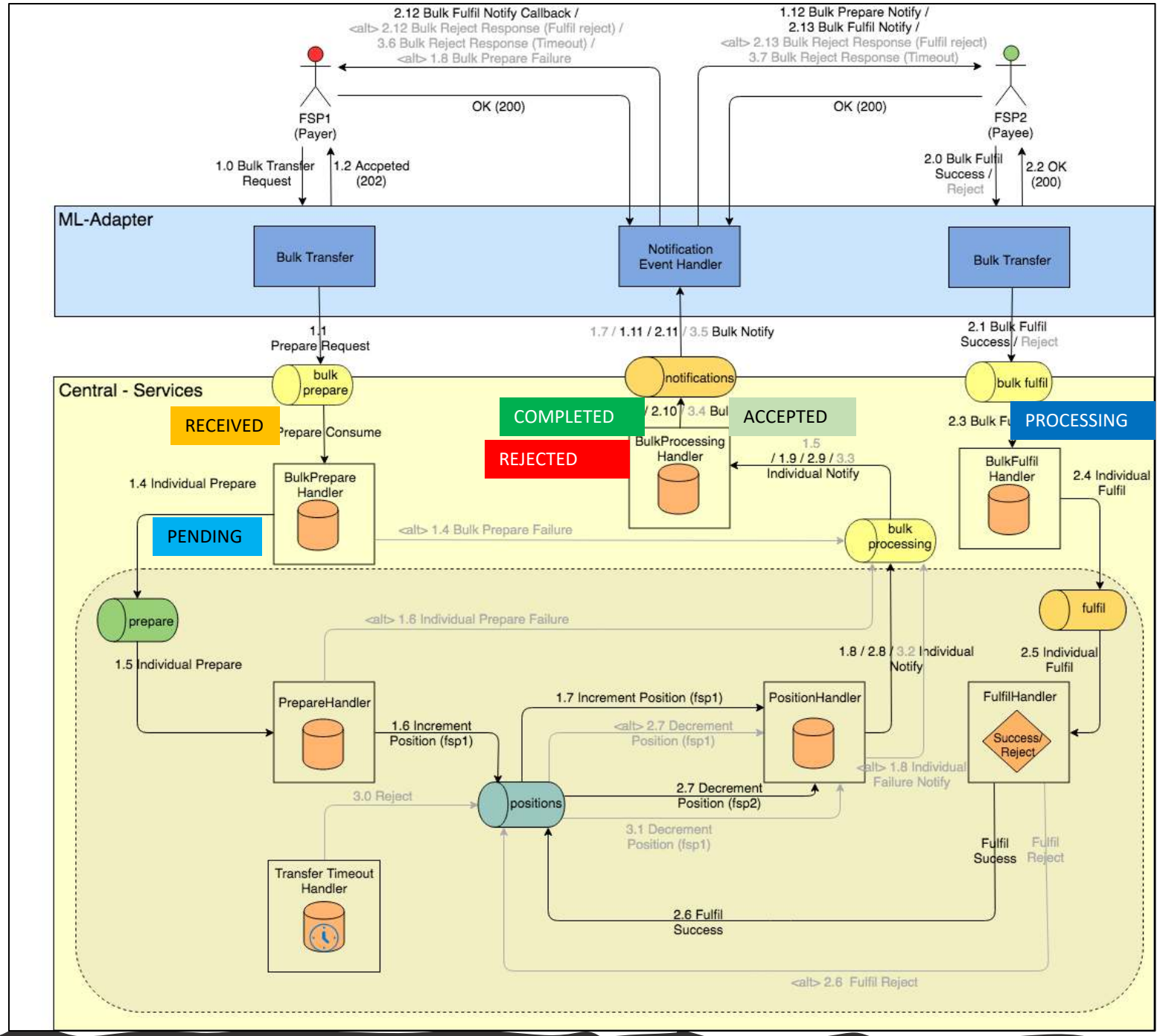
PI-5: Bulk Transfers – Notes, Changes

1. Switch to validate bulk, break down into individual transfers
2. Ordering of the individual transfers – Maximize # of transactions?
3. Send out Bulk Prepare to Payee only for reserved transfers
4. Send a single aggregated notification (PUT callback) to payer
5. Address considerations for GET calls, validation
6. Validation failures on bulk - **Rejected**
7. Even a single individual transfer is committed - **Completed**

PI-5: Bulk Transfers States



PI-5: Bulk Transfers High Level Architecture



PI-6

PI-5 Bulk Transfers: Design Considerations

1. Specification Changes to allow Switch capabilities
2. Impact on Settlements, changes needed to support G2P - Discuss
3. Time-outs considerations
4. Size considerations of bulkTransfers messages
5. Headers, Signature/Encryption, Security considerations
6. Ensure correct behavior for GET on /bulkTransfers/{ID} and changes proposed
7. Need for a bulk-make sort of a feature – for the Switch to take over major processing with lookups, quotes?

Open PRs for Design:
https://github.com/mojaloop/docs/pull/143
https://github.com/mojaloop/documentation/pull/29
https://github.com/mojaloop/central-ledger/pull/267

PI-5 Bulk Transfers: Roadmap

1. Incorporate feedback from discussions, document
2. Finalize design for PoC
3. Implement PoC
4. Submit changes to the API to the CCB
5. After resolution, complete implementation

Settling liabilities in Mojaloop systems

Overview and discussion

How are liabilities incurred?

1. When funds are transferred, the payee DFSP actually credits the beneficiary.

2. Those funds can be cashed out, or converted into any other form at the discretion of the beneficiary…

3. But the payee DFSP hasn't yet received anything more than the promise of payment from the payer DFSP.

4. The promise to pay is the payer DFSP's *liability* to the payee DFSP

The motto of settlements

"Jack shall have Jill;
Nought shall go ill;
The man shall have his mare again,
and all shall be well."

William Shakespeare

A Midsummer Night's Dream

III, ii, 490-493

Liability and confidence

1. How can a participant be confident that other participants in the scheme will make good on their liabilities?
2. One way is to ensure that no participant can incur liabilities that they are unable to settle.
3. All liabilities are backed by actual funds in a Settlement Account; and perhaps additionally by collateral or credit arrangements at the Settlement Bank
4. Participants are not allowed to transact if their liabilities aren't covered by one of these methods
5. Which implies that, for each transaction, the current total liability of the payer DFSP is calculated and checked against its ability to settle its liabilities
6. A scheme's assessment of a participant's ability to settle its liabilities is defined as the participant's *Net Debit Cap*.
7. *But we're still in the land of promises…*

Net Debit Cap: Manual vs. Automated

1. The Net Debit Cap (NDC) prevents participants incurring liabilities that they may not be able to settle.
2. Its base position is the participant's balance in their Settlement Account.
3. The base position may be varied
4. Positively, for instance to encourage participants or to express trust…
5. Negatively, for instance to allow for account unavailability
6. Ideally, the scheme should be able to get direct information about the participant's balance in their Settlement Account, and use this to set the participant's NDC automatically.
7. In any case, a scheme needs to be able to set a permitted variance margin per participant, or for the scheme overall.
8. If the NDC is to be reliable, the Scheme must be able to approve the removal of funds from the Settlement Account and prevent situations where funds are removed from the Settlement Account between the time the liability is incurred and the time it is settled.

Settlement is the process of making those promises real

1. Which means: each participant should be able to possess the funds they're owed and should be required to disburse the funds they owe.

2. This discussion is about how that happens

3. Next, we look at the sorts of choices that need to be made.

Types of Settlement: Bilateral or multilateral settlements

Bilateral settlements

1. In bilateral settlements, each participant settles separately with every other participant with which it has participated in a transfer.

2. This can be done on either a net or a gross basis

3. That’s a lot of settlements…

4. But it *does* mean that settlement is actually occurring between the participants who incurred the liabilities

Types of Settlement: Bilateral or multilateral settlements

Multilateral settlements

1. In multilateral settlements, each participant settles once for all the transfers it has participated in, no matter who the counterparty was.
2. That's a lot fewer settlements
3. But it means that settlement is *actually* occurring between the participant and the scheme, not between the participants who actually contracted the liabilities.
4. In multilateral settlements, the scheme is guaranteeing the settlement of liabilities
5. This is (by far) the most common type of settlement in implemented schemes

Types of Settlement: where is settlement made?

1. Settlement is made using a physical bank account

2. The types of bank account which might be involved are..

Types of settlement bank account

1. Settlement bank accounts can be hosted by
   a. A central bank
   b. A commercial bank
2. Settlement bank accounts can be
   a. Maintained separately for each participant
   b. Grouped together, either
      i. Where all settlements take place in a single account
      ii. Where some participants maintain accounts on behalf of (themselves and) others.
         A. Where this occurs, it is good practice for the scheme to ensure that the positions of all participants who use the shared account can be treated separately
   c. Or a mixture of the two…
3. Settlement bank accounts can be
   a. Used only for the purposes of settlement
   b. Used for other purposes as well
      i. In which case, control of the accounts for the purposes of ensuring that settlements can always be covered becomes problematic
4. What is *technically* possible may be limited by legislation or regulation in particular jurisdictions

How do we record the list of promises that were made?

1. The Scheme maintains an internal Position Ledger for each participant (and for each currency in which a participant transacts, where more than one currency is supported.)

2. Each funds transfer in which the participant is involved is recorded in the ledger
   a. If the participant is the payer, the ledger is debited.
   b. If the participant is the payee, the ledger is credited.

3. It's important to remember: whether the settlement model is net or gross, this is a question of liquidity
   a. The scheme needs to know when a ledger entry has been marked for settlement.
   b. The scheme needs to know when the participant's Settlement Account has been adjusted to settle that ledger entry.

Types of Settlement: Net Settlement

1. *How it works*: at a specified point, the net position of every Position Ledger over the *net settlement period* (i.e. since the last time a net settlement was completed) is calculated.

2. This net position excludes any transfers which have been requested but not yet completed.
   a. When a transfer is requested, the amount of the transfer is reserved in the payer's ledger, but nothing is yet recorded in the payee's ledger.
   b. So the ledger position is asymmetric with respect to incomplete transactions.

Net Settlement (continued)

1. How it works: at a specified point, the net position of every Position Ledger over the *net settlement period* (i.e. since the last time a net settlement was completed) is calculated.

2. If the net amount shows more debits than credits:
   a. The participant owes money to the scheme
   b. The Scheme deducts funds from the pre-funded total in the participant's Settlement Account.
   c. It is the participant's responsibility to transfer funds into the Settlement Account to restore their pre-funded total and allow them to continue operating

3. If the net amount shows more credits than debits:
   a. The scheme owes money to the participant
   b. The funds are added to their Settlement Account's pre-funded value
   c. The participant may withdraw some or all of the funds from their Settlement Account, subject to the consent of the Scheme

Types of Settlement: Gross Settlement

1. *How it works*: the Settlement Accounts for each participant in a transfer are adjusted for each transaction that is completed, as it is completed
2. Settlement amounts and pre-funded values are always up to date.
3. This works very efficiently for a single pooled account, but potentially requires an account transfer mechanism which can support large numbers of transactions if a single pooled account is not used.
4. This is how RTGS works for bank reserve accounts
   a. But RTGS is designed for high-value, low-volume transactions
   b. This model may not be appropriate for the volumes of transactions we have in mind.
5. The single pooled account is analogous to the way in which Mobile Money systems work successfully today:
   a. The pooled account is equivalent to the Control Account in a Mobile Money system: it represents the total amount of funds in the system.
   b. The Scheme, like the MMS, records the position of individual participants in relation to the total account
   c. Participants can add funds to the account as required, in the same way as participants in an MMS can add funds to their e-money accounts by adding funds to the Control Account.

Net vs. Gross Settlement: Pros and Cons

1. Accounts can be either pooled or individual
   a. Pooled accounts can cover a combination of participants, while other participants have their own accounts;
   b. Or there may be a single account which pools the funds of all participants

2. Individual accounts can be either dedicated or commingled
   a. Commingled accounts are accounts which are, or can be, used for purposes other than settlement.
   b. Where an account can be used for purposes other than settlement:
   c. it is not possible to rely on the funds in it for the purposes of assessing the liabilities which a participant can guarantee.
   d. The Net Debit Cap for the Participant(s) who use this account must therefore be set manually by the Scheme

This gives us the following settlement type matrix

Problems and constraints in various settlement models			
Settlement Model	Individual Participant Settlement Bank Account, Multi-Purpose	Individual Participant Settlement Bank Account, Scheme-Dedicated	Pooled Settlement Bank Account (all Participants), Scheme-Dedicated
Net Settlement			
Gross Settlement			

Net vs. Gross Settlement: What are the potential problems?

1. All commingled accounts prevent the automation of the Net Debit Cap

2. All gross settlement to individual accounts has the RTGS volume limitation problem
    1. RTGS systems are designed for high-value, low-volume transactions
    2. Our model is low-value, high-volume…

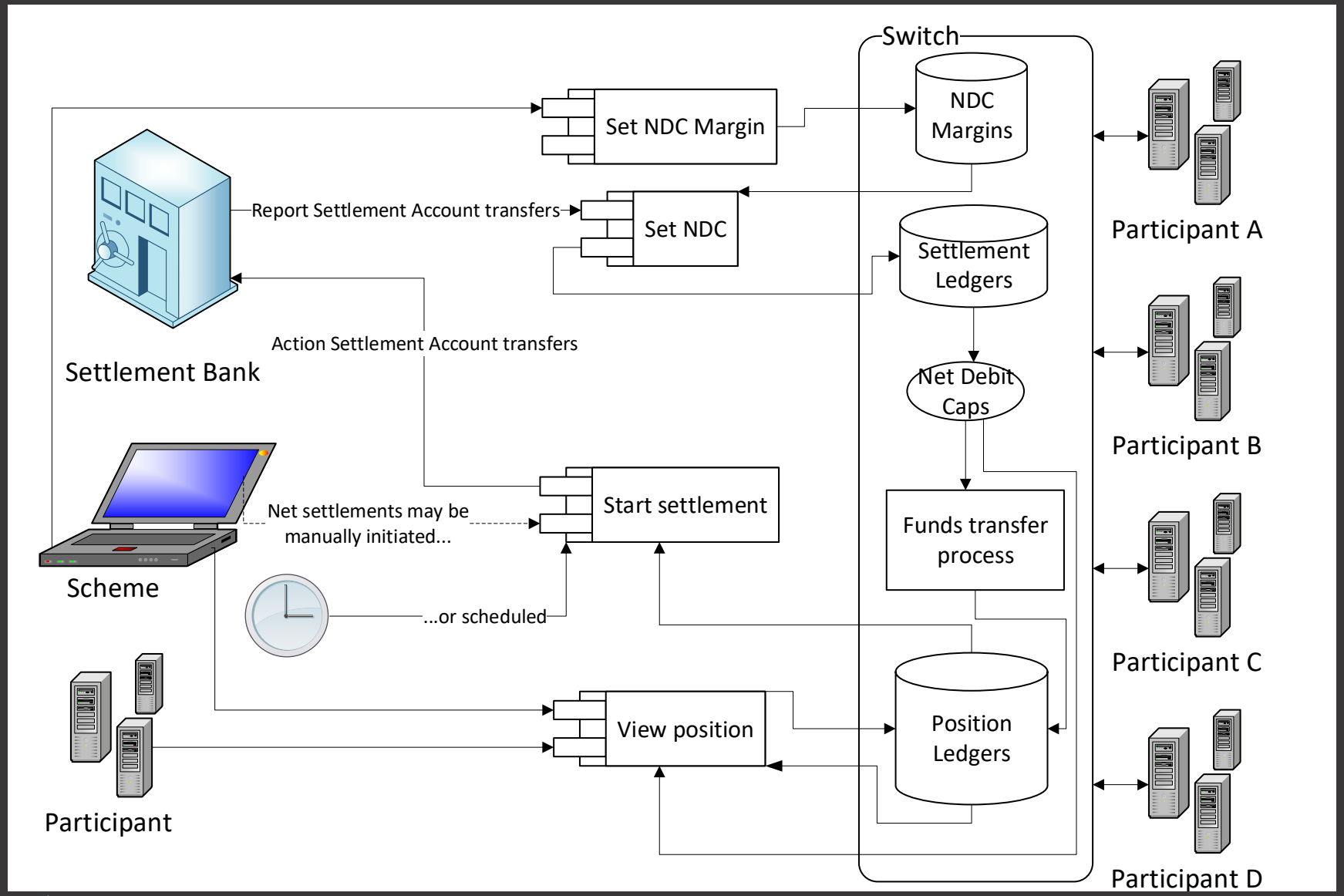
3. All net settlement to individual accounts has the 24x7 problem
    1. Net settlements will typically require larger adjustments of funds…
    2. Systems may not be available when required

4. All net settlement has the reconciliation problem
    1. Because settlement takes place in large batches, participants can only reconcile after the fact.
    2. Whereas gross settlement allows participants to check their expected positions in real time.

This gives us the following settlement type matrix

Problems and constraints in various settlement models			
Settlement Model	Individual Participant Settlement Bank Account, Multi-Purpose	Individual Participant Settlement Bank Account, Scheme-Dedicated	Pooled Settlement Bank Account (all Participants), Scheme-Dedicated
Net Settlement	• No automated NDC calculation • 24x7 problem • Reconciliation problem	• 24x7 problem • Reconciliation problem	Why would you?
Gross Settlement	• No automated NDC calculation • High volume constraint	• High volume constraint • 24x7 problem	

Suggested outline API Schematic for settlements



The floor is yours…

mojaloop

Fraud & AML

Supporting Adoption & Deployment

Requirements now available for feedback

- Document now available here
- Key components Identified

1. Risk Scoring of Transfers
2. Process Modelling
3. User Blacklisting
4. Data Repository
5. Transaction Tracing
6. Key Event Monitoring
7. Management Reporting
8. Suspicious Activity Reporting
9. Case Management
10. External Data Capture

Key Tasks we need to address

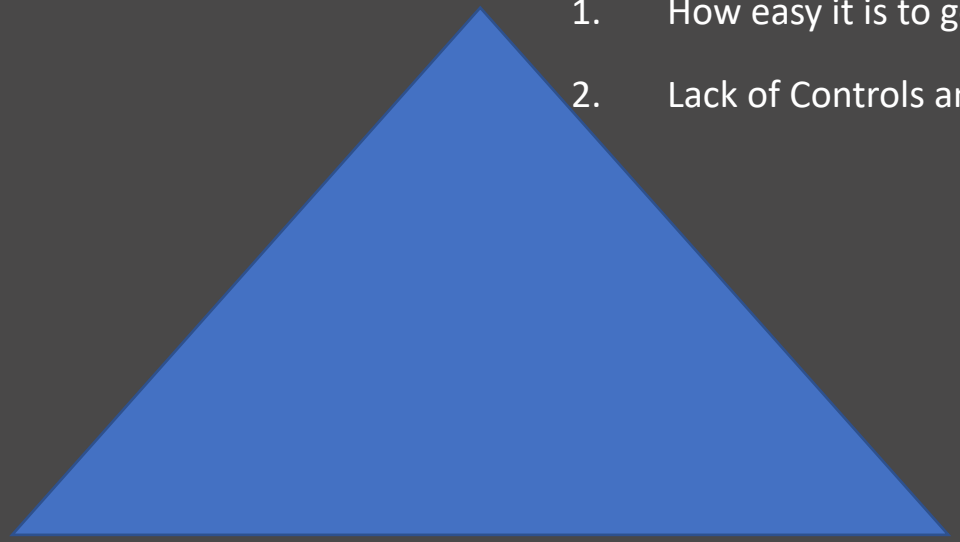
1. We need to introduce some new processes to assist in the sharing of typologies
    a. A common framework for creation
    b. Management of who sees the fraud type
    c. Management of who sees the resolution
2. Whilst maintaining an Open Source approach

Cressey's Fraud Triangle

Main drivers for Fraud

Opportunity

1. How easy it is to get away with the act
2. Lack of Controls and Oversight



Rationalisation

1. Victimless crime
2. Propensity to commit crime

Pressure

1. Financial Security
2. Coercion

Gated Milestone Approach

1. Adopted from the Level One Project

   Fraud Scenarios in the Mobile Money Ecosystem

2. Will assist in the identification of weak points

3. Will be used when analyzing the Typology

Prospecting 1
Identification of accounts where money can be stolen from

Norming 2
Creation of the Fraud Execution Process – but not limited to
- Removal of funds from identified Account
- Parking of the funds
- Liquidation of funds

Transacting 3
Fraud executed and Financial Transition is performed

Liquidation 4
Money is liquidated into cash or services

Assessment of the Typology

STRIDE – used for Security threats

1. Spoofing Identity
2. Tampering with Data
3. Repudiation
4. Information Disclosure
5. Denial of Service

DREAD – used for assessing threat risk Level

1. Damage Potential
2. Reproducibility
3. Exploitability
4. Affected Users
5. Discoverability

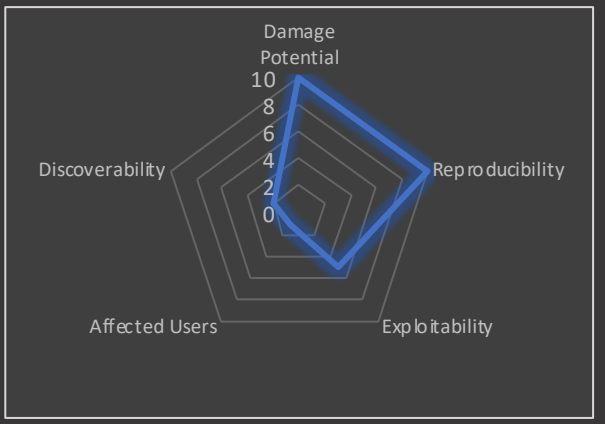
Sample Typology – that would impact settlement

Spoofing Emails of the Bank Mandate Holder

1. Transfer account for Funds Out changed via email notification
2. When DFSP Requests Excess Funds to be transferred back, they are sent to the new account – and rapidly moved on

Stride Classification	Stage of Fraud Assistance
1. Spoofing	1. Financial Transaction
2. Denial of Service	

DREAD SCORE	5.6
Damage Potential	10
Reproducibility	10
Exploitability	5
Affected Users	1
Discoverability	2

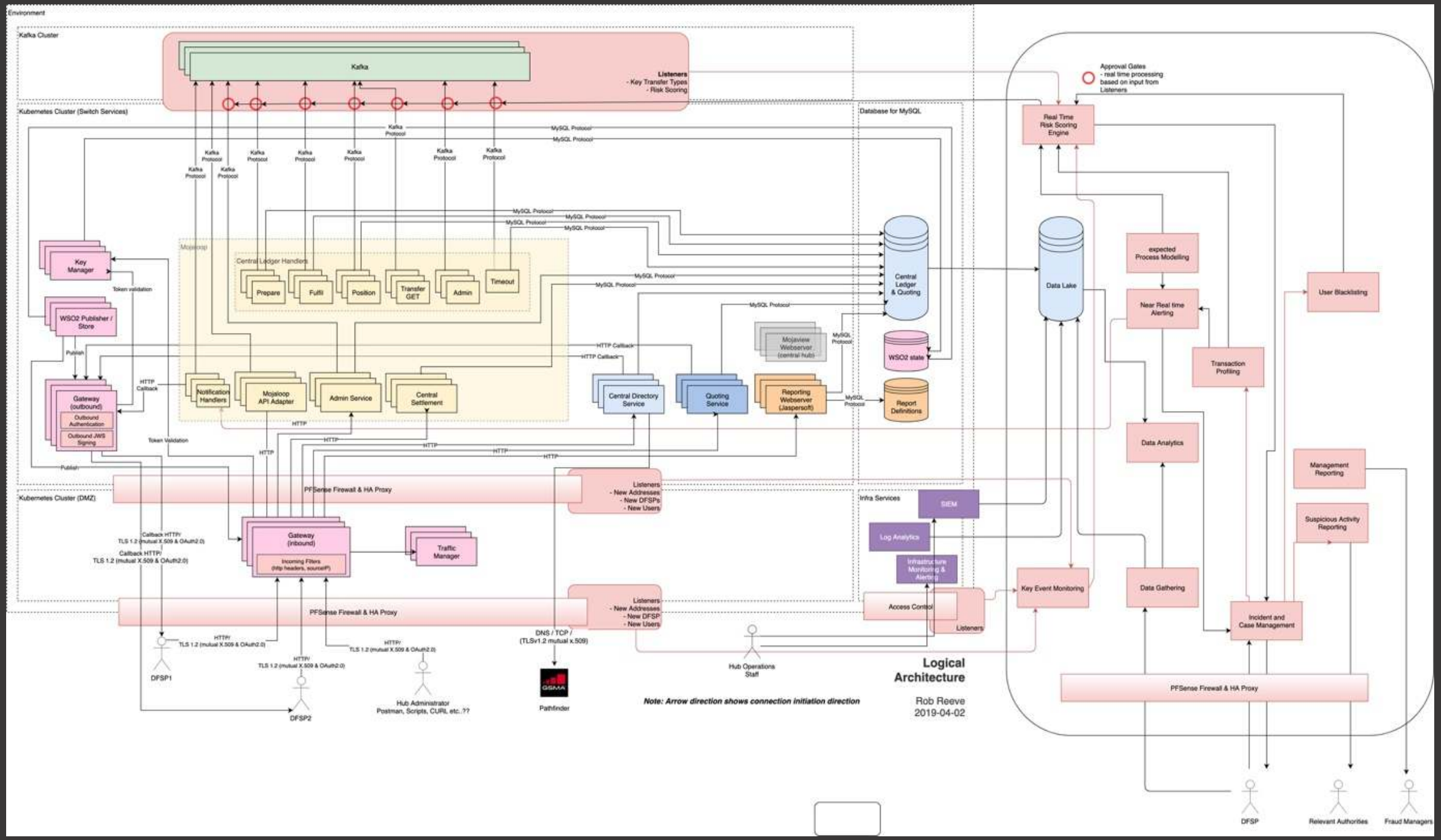


Sharing of Typologies

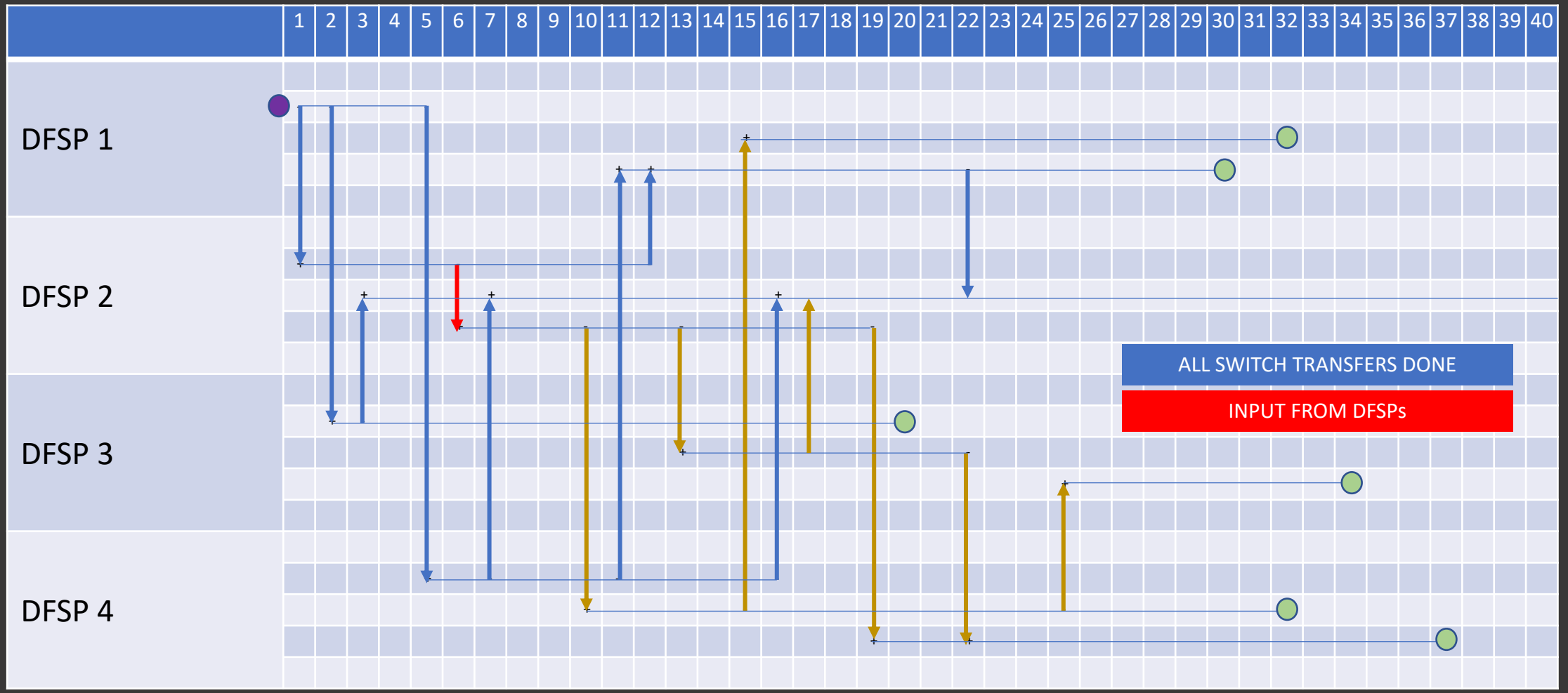
1. During Assessment
    a. We risk sharing new approaches in Fraud with the criminals who we are looking to stop – before a remediation has been implemented
2. Post implementation
    a. We need to prove the development works – before the Typology is released

3. But there are examples of this process working well already for Fintechs

Architecture



Money Tracing as an Example

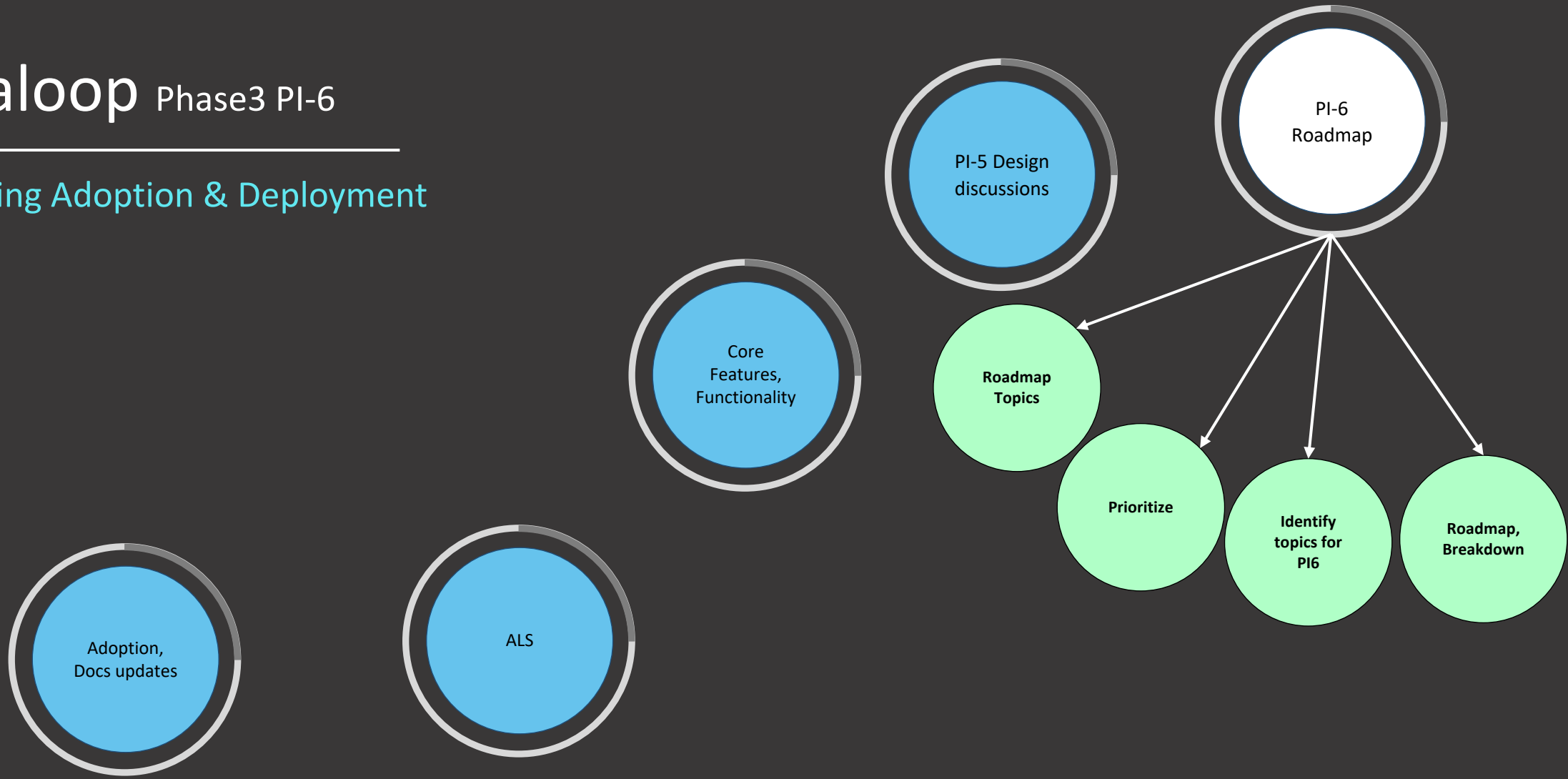


Working Assumptions on implementation

1. Case and Incident Management will not be addressed – most organizations already have an existing platform
    a. But Open Source solutions should be suggested for those who do not
2. Machine Learning and Modelling will be introduced – however data would be required before any real value would be seen
    a. This moves into the realm of Data Scientists
3. This will be revisited in the PI Planning

Mojaloop Phase3 PI-6

Supporting Adoption & Deployment



16th – 18th April 2019

Switch Functionality – Mojaloop End-points (Goal PI-6)

Mojaloop v1.0 – API Specification

Transfers
[●] POST - Prepare
[●] PUT - Response
[●] PUT – Error
  [●] Outgoing
  [●] Incoming
[●] GET - Query

Parties#
[●] GET - Request
[●] PUT - Response
[●] PUT - Error

Quotes*
[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[●] GET - Query

Participants#
[●] POST - Create
[●] PUT - Response
[●] POST - Bulk Create
[●] PUT - Error
[●] DEL - Delete

Transactions
[○] PUT - Response
[○] GET - Query

TransactionRequests$
[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[●] GET - Query

- * Lightweight quoting service for relaying
- $ Lightweight service similar to quotes
- % Finalize design for V1 and PoC
- # Complete current effort

Authorizations
[○] GET - Request
[○] PUT - Response
[○] PUT - Error

BulkTransfers%
[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[●] GET - Query

BulkQuotes
[○] POST - Request
[○] PUT - Response
[○] PUT - Error
[○] GET - Query

Key
[●] Fully implemented
[●] Legacy Code
[●] Partially implemented
[●] Not implemented
[○] Out of Scope for PI6

PI-6 Roadmap: Continuation of ongoing features

1. Cross network [Coil]
2. Cross currency [Coil]
3. Account Lookup Service – Oracle Template
4. Bulk Transfers (Goal - PoC, change proposal)
5. Documentation (move over everything from 'docs' repo – Support for languages)
6. QA, Testing

PI-6 Roadmap: New Topics

1. Error Handling
2. Event Handling Framework
3. Forensic Logging Sidecar
4. Merchant Payments
5. Fraud & AML ?
6. API Gateway
7. Settlements API
8. Monitoring Production Based systems

PI-6 Roadmap: Other Topics

1. Demonstration tools - such as USSD, etc
2. Licensing - start the process of assessing current state and planning next steps also specific items for items that can be easily addressed
3. Quality (Revive SonarQube) - Fix high priority issues
4. Quotes service? (Use lightweight quoting service, explore interop-switch-js by Coil)

PI-6: Objectives

1. An Error & Event handling Framework is implemented that will support Forensic Logging
2. A comprehensive account lookup service and a template which can be used to create a Pathfinder implementation
3. The 'documentation' repo is made up-to-date and 'docs' repo deprecated
4. Bulk Transfers design is finalized for V1.0 and a PoC is implemented; Changes proposed to CCB
5. A native quotes service is incorporated (possibly Coil)
6. API Gateway (WSO2) is implemented
7. A Settlements API for OSS is drafted and published
8. Quality metrics - SonarQube is live and enabled for all core services and bugs logged for high priority issues reported
9. Optimistic: Customer Initiated Merchant Payment use case is supported (Merchant lookup - use 'Payment ID' proposal)
10. Optimistic: Support for a demonstration tool for a P2P use case; Utilization of an SDK when ready

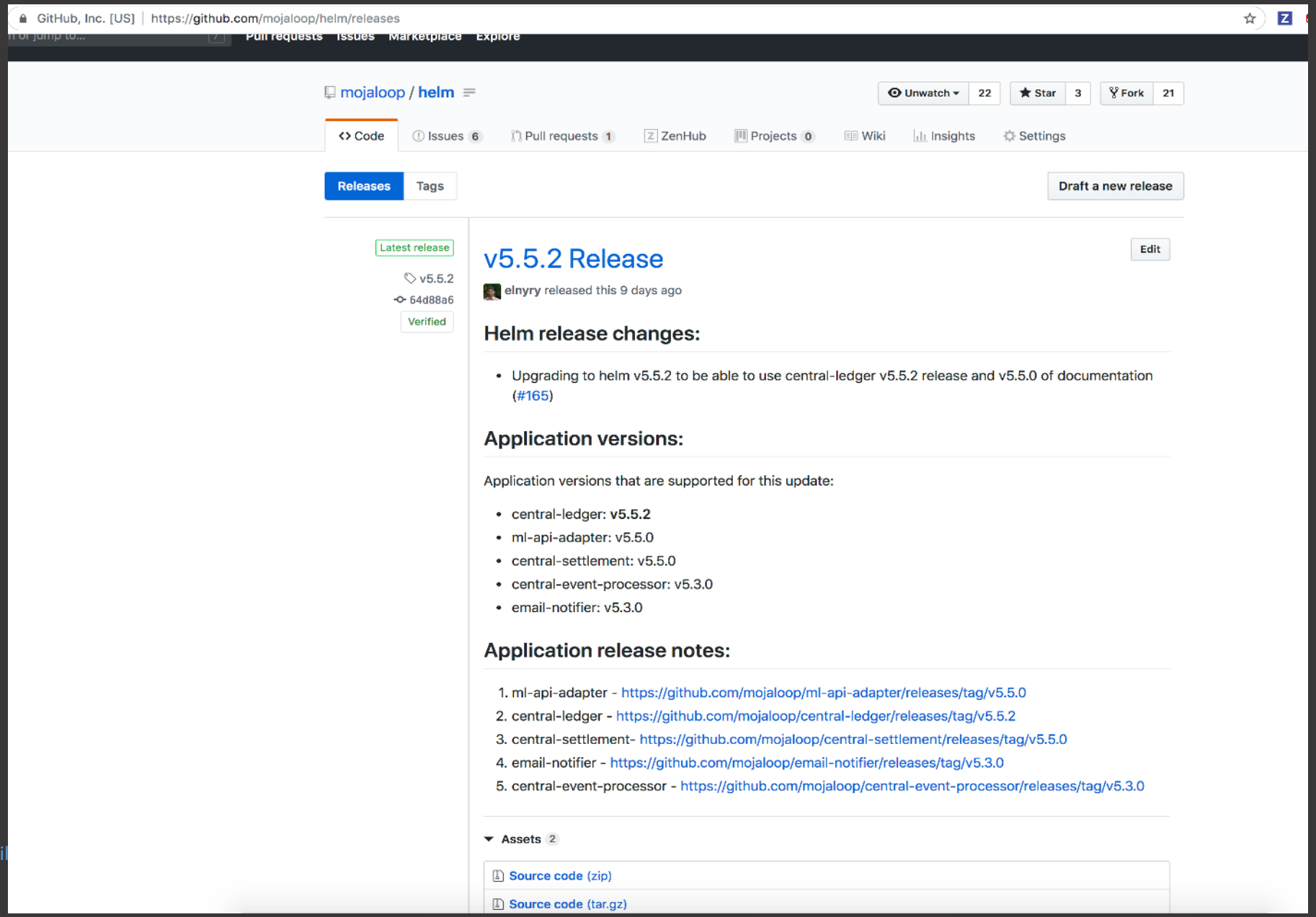
PI-6: Dependencies

1. Fraud is dependent on the Event Framework
2. Coil dependency on quoting service
3. API Changes dependent on the CCB – affects ALS, Cross currency/border, Quotes

PI-6: Risks

1. Risk regarding API changes (proposed by Coil)
2. Risk regarding Scope (For Objectives #8 onwards)
3. Data generation for Fraud

Release Mechanism: Sample Helm Release



Go Back

Release Mechanism: central-ledger Release Notes

https://github.com/mojaloop/central-ledger/releases

v5.4.0
122a9ed
Verified

v5.4.0 Release

Edit

mdebarros released this 25 days ago · 5 commits to master since this release

Bug fix changes:

- Fixes below cater for the following bug mojaloop/project#687
- POST /transfers when fulfil transferState='ABORTED' now returns the correct payload
- GET /transfers when fulfil transferState='ABORTED' now returns the correct payload
- Minor bug in the Participant domain was not returning the headers as the message contained `header` instead of `headers` key
- GET /transfers was not returning the correct error when a transfer ID was not found. This now returns `ERROR 3208`.
- Central-switch designation has been changed to `switch` which is aligned to the Mojaloop specification

Maintenance changes:

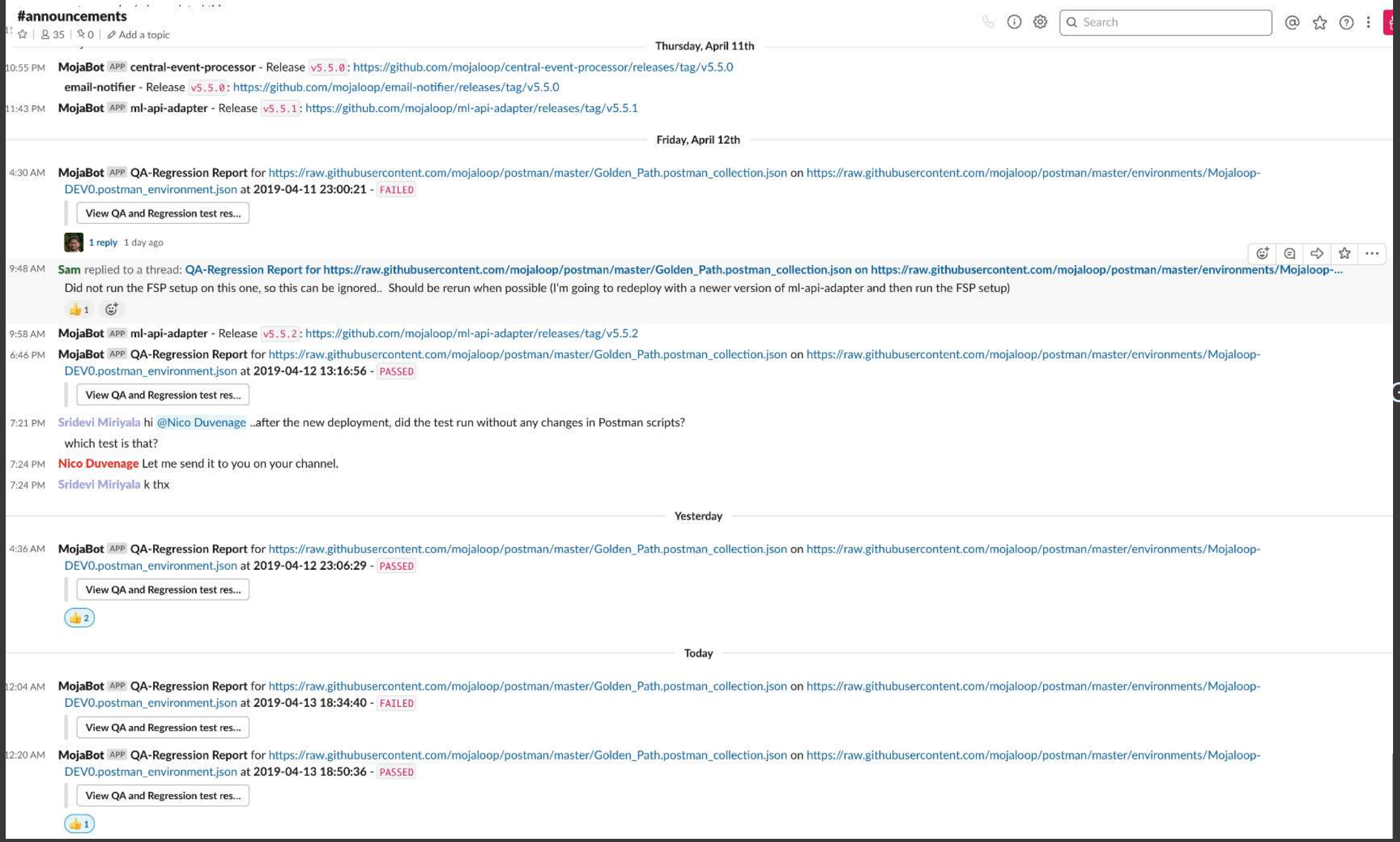
- Added a new Util.clone() method and updated most (if not all) clone methods to use this common operation
- Moved toFulfil transform method from the Transfer Handlers to transformer.js
- Added unit tests for the toFulfil transformer method to ensure code-coverage is maintained
- Added new ENUM for TransferStateEnum which is required for logical comparison within the Transfer Handlers
- Updated all Transfer Transformers to return only fields that have values, and leave out any undefined/null key-values from the output transformation result in transformer.js
- Bumped version to 5.4.0
- Cleaned up config by removing kafka configs for reject and abort as their respective handlers do not exist.

▼ Assets 2

Source code (zip)
Source code (tar.gz)

Go Back

Release Mechanism: Slack Announcements



Go Back