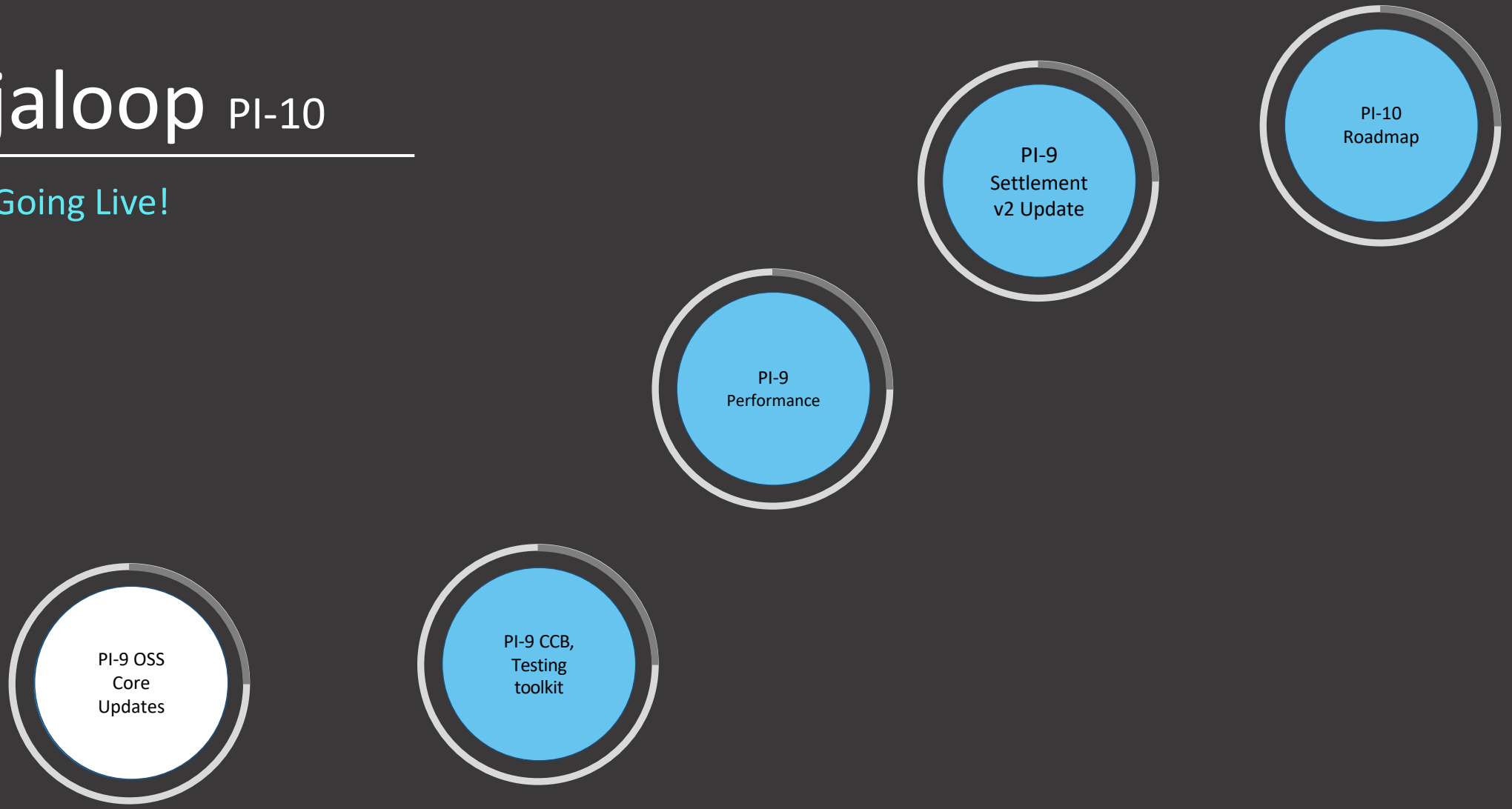
Mojaloop PI-10

Phase4 Going live!

Mojaloop PI-10

Phase4 Going Live!



PI-9 Overview: Agenda

1. PI-9 Overview of changes and progress
2. OSS Community, Collaboration - Updates
3. PI9 Focus Areas

Mojaloop PIs Overview

Timeline	Summary
Phase-1 (2016 - 17)	**Level One Project** • Reference Implementation • 6 Program Increments (PIs)
Phase-2 (2018)	**Road To Productionization** • 1 – 4 Program Increments
Phase-3 (2019 Jan - Dec)	Supporting Adoption & Deployment • PI-5 (Feb – April): Account lookup, QA Framework, Streamlined CI, Release process, Error endpoints, Documentation, Node Upgrade, Bug Fixes & Community support, Bulk Transfers Design • PI-6 Event handling framework, Bulk Transfers PoC, API Gateway, OSS Settlements API, Quoting Service, ALS • PI-7 Event & Error Handling framework, Packaging, OSS Settlements, Performance testing capabilities, QA • PI-8 Consolidation, Performance, Community Support
Phase-4 (2020 Jan -)	**Going Live** • **PI-9:** Maintenance, Performance Testing & Improvements, Merchant Request to Pay, Operational Monitoring, Testing toolkit, Settlement v2 • **PI-10** (May – July) • PI-11 • PI-12

Switch Functionality – Mojaloop Use Cases (Phase-4 PI19)

Mojaloop v1.0 – Use-cases

Payer-Initiated Transaction

[●] P2P Transfers
- [●] Prepares, Fulfils
- [●] Rejections, Timeouts
- [●] Error Endpoints

[●] Customer-Initiated Merchant Payment
[●] Customer-Initiated Cash-out - Receive Amount
[●] Customer-Initiated Cash-out - Send Amount
[●] ATM-Initiated Cash-out
[●] Refund

Bulk Transactions

[●] Bulk Payments

Payee-Initiated Transaction

[●] Merchant-Initiated Merchant Payment
[●] Agent-Initiated Cash-out
[●] Agent-Initiated Cash-In – Send Amount
[●] Agent-Initiated Cash-In – Receive Amount

Payee-Initiated Transaction using OTP

[●] Merchant-Initiated Merchant Payment Authorized on POS
[●] Agent-Initiated Cash-out Authorized on POS

Key
[●] Fully implemented
[●] Supported, not tested
[●] Proof of Concept
[●] Not implemented
[○] Out of Scope

Switch Functionality – Mojaloop (Phase-4 PI19)

Mojaloop v1.0: Focus Use-cases

1. P2P
2. Merchant 'Request to Pay'

Payer-Initiated Transaction

[●] P2P Transfers
- [●] Prepares, Fulfils
- [●] Rejections, Timeouts
- [●] Error Endpoints

Payee-Initiated Transaction*

[●] MIMP Transfers
- [●] Transaction Requests
- [●] Prepares, Fulfils
- [●] Rejections, Timeouts
- [●] Error Endpoints

Mojaloop v1.0 – End Points for P2P

Transfers

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

Quotes

[●] POST - Request
[●] PUT - Response
[●] PUT - Error
[●] GET – Query

Authorizations*

[●] GET - Query
[●] PUT – Response
[●] PUT - Error

Parties

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

Participants

[●] POST - Create
[●] PUT - Response

TransactionRequests*

[●] POST - Request
[●] PUT – Response
[●] PUT - Error
[●] GET - Query

Key
[●] Fully implemented
[●] Legacy Code
[●] PoC / Initial Version
[●] Partially implemented
[●] Not implemented
[○] Out of Scope

* Enhancements for PI9

Switch Functionality – Mojaloop End-points (PI8→PI9)

Mojaloop v1.0 – API Specification

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

1. **Performance improvements**

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

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
[●] PoC / Initial Version
[●] Partially implemented
[●] Not implemented
[○] Future Roadmap

* Enhancements for PI9

PI9 Switch Functionality – Operations

Operational – Use Cases

Participants

[●] Manage Participants
- [●] Create Initial Value
- [●] Query
- [●] Update

[●] Manage Participant Limits
- [●] Create Initial Value
- [●] Query
- [●] Update

[●] Manage Callback URLs
- [●] Create Initial Value
- [●] Query
- [●] Update

Oracles

[●] Manage Oracles
- [●] Create
- [●] Query
- [●] Update
- [●] Delete

Settlements v**1.0** [Supports some forms]

[●] Open, close Settlement Windows
[●] Query Settlement Windows
[●] Query Settlement Report
[●] Create/Trigger Settlement with Windows
[●] Process successful Settlement Acknowledgements
[●] Reconcile Positions based on successful Settlements
[●] Process failed Settlement Acknowledgements

Positions

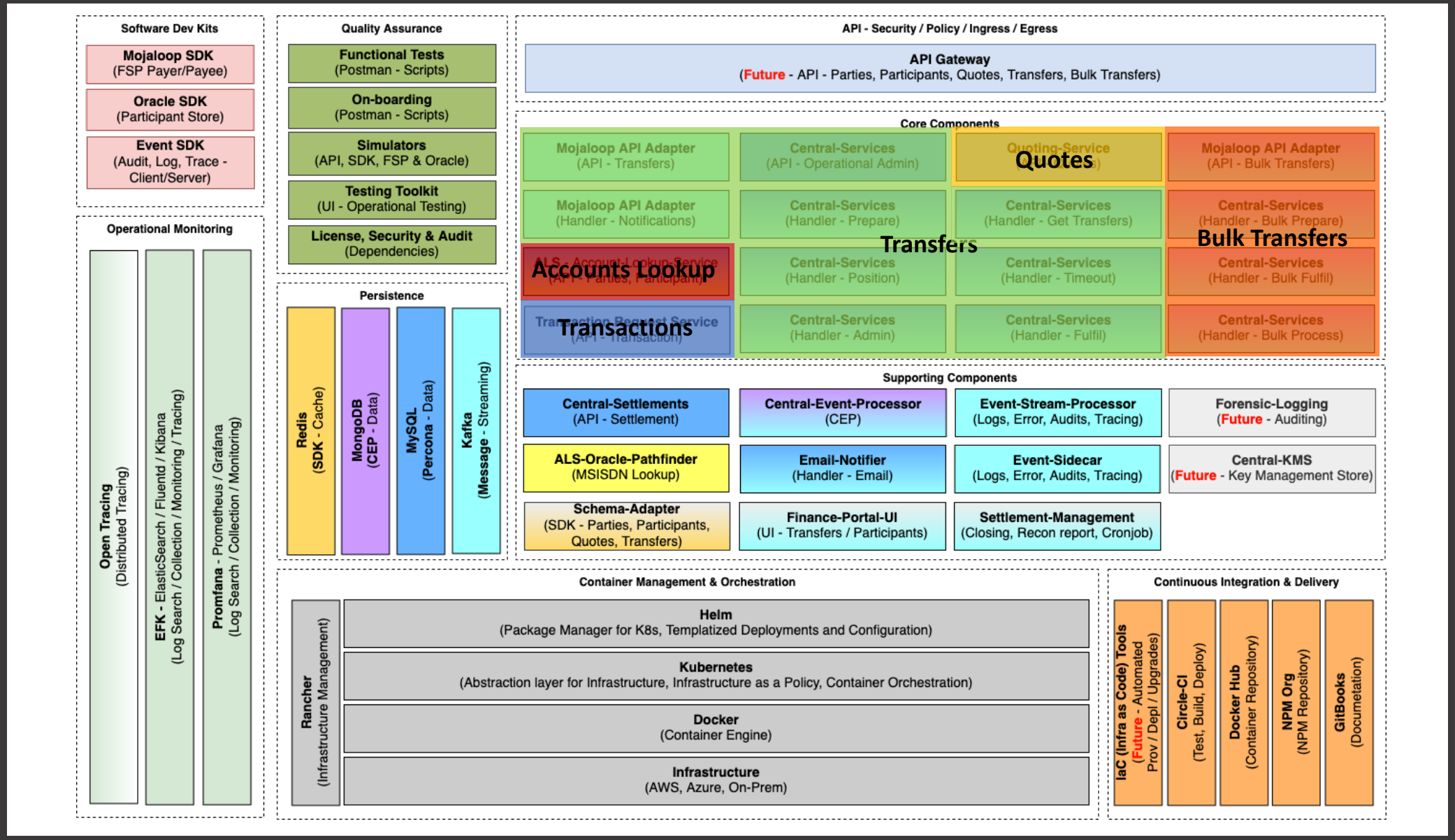
[●] Query Positions
[●] Manage Positions
- [●] Create Initial Value
- [●] Query
- [●] Update

Settlements v**2**

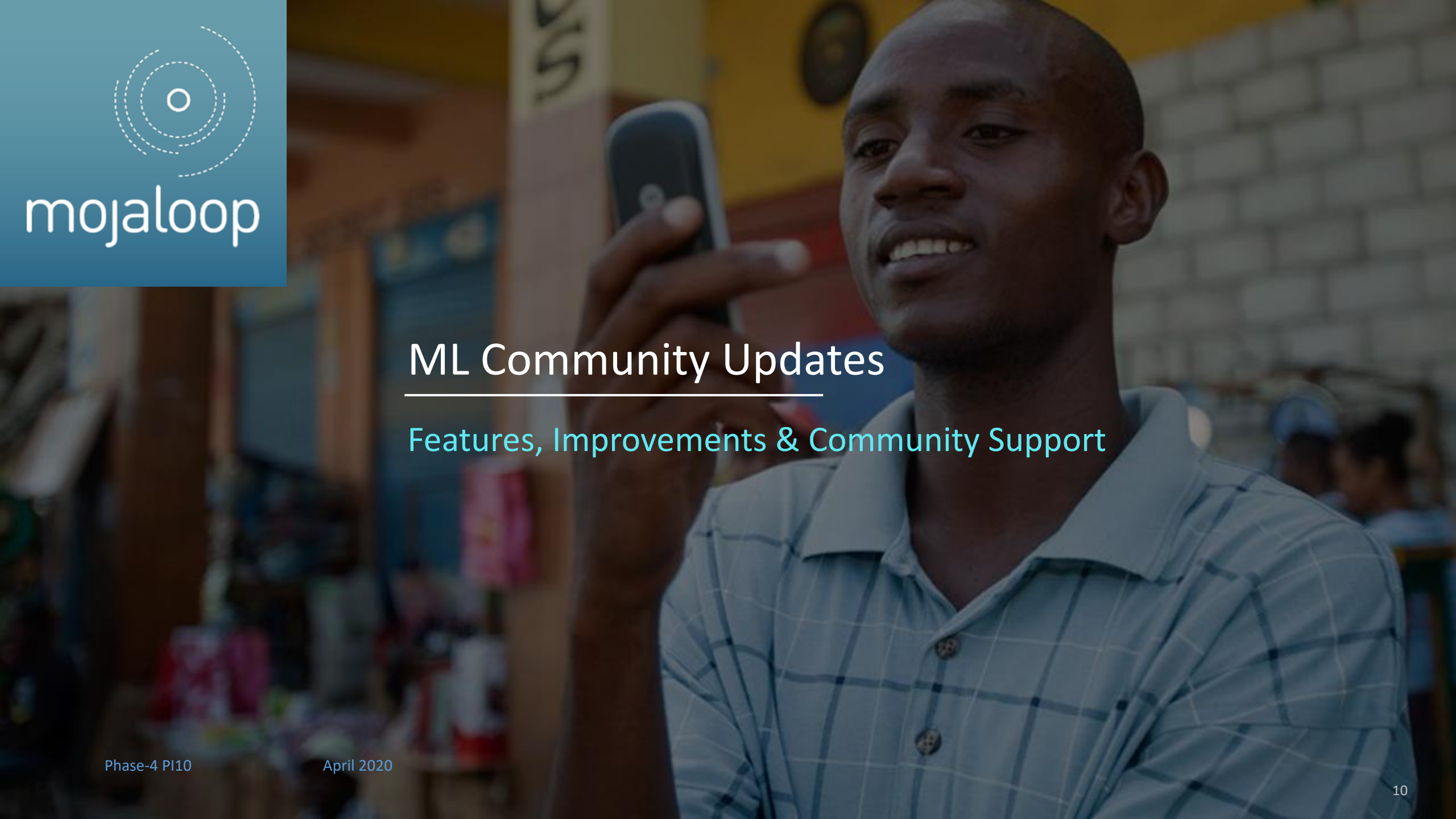
[●→●] Settlement models
[●→●] Designs
[●→●] Settlement by Currency
[○] Interchange Fees
[○] Continuous Gross Settlement

Key
[●] Fully implemented
[●] Legacy Code
[●] PoC / Initial Version
[●] Partially implemented
[●] Not implemented
[○] Roadmap for PI10

PI-9: Component Architecture



* Color-shading of core & supporting components match dependent persistent stores.



ML Community Updates

Features, Improvements & Community Support

PI-9 Overview: Mojaloop OSS Community

Contributions, Collaboration

1. Cross network/currency
2. Code quality and Security improvements
3. Versioning Standards
4. ISO Integration
5. Quality Assurance (bugs, coverage, Tests with mojaloop-simulator)
6. Settlement v2
7. PISP Solution

Mojaloop OSS PI9: Community

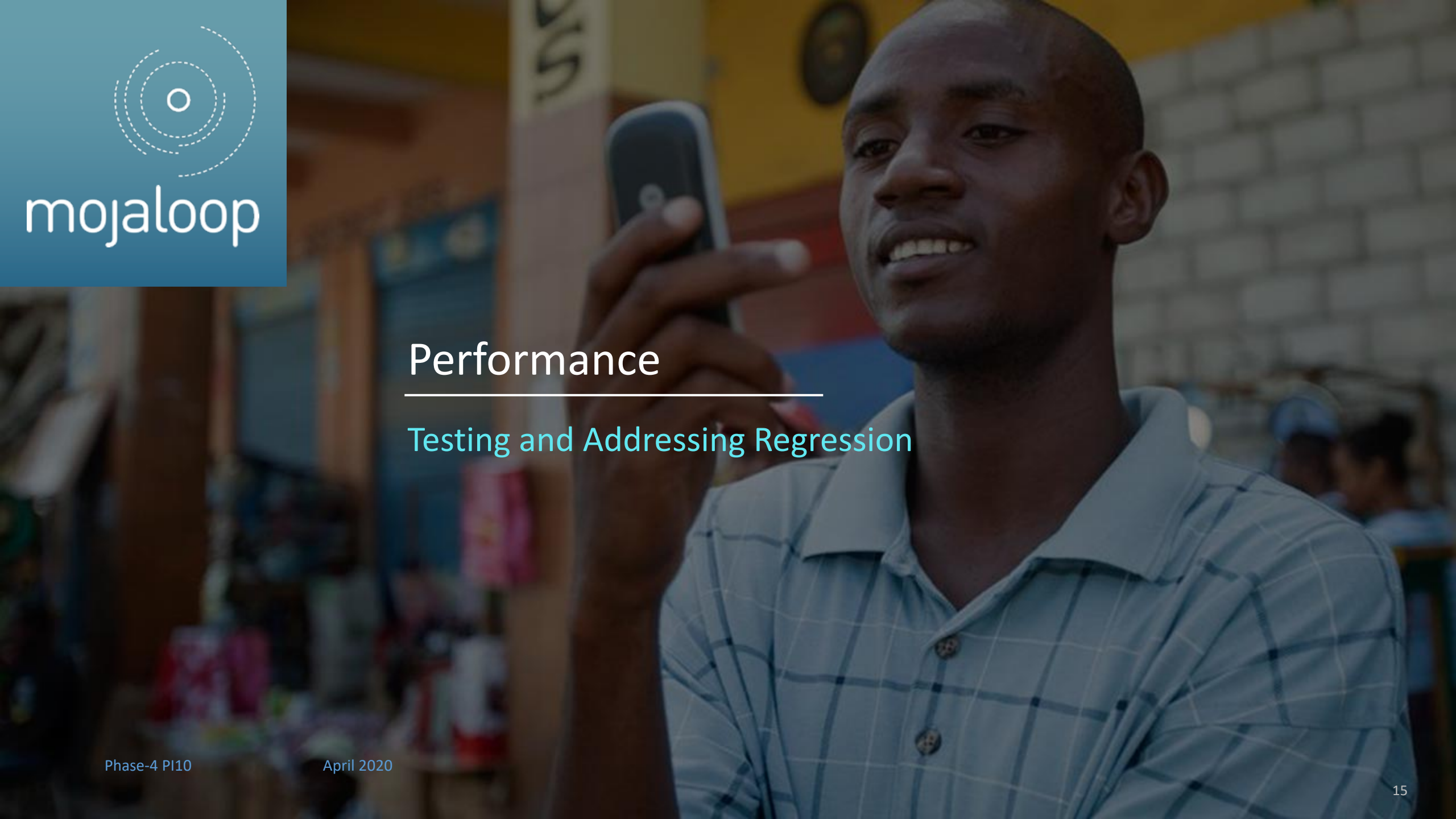
1. Change Control Board (CCB)
2. Design Authority (DA)
3. Weekly scrum-of-scrums
4. Slack channels #*general*, #*announcements* , #*help-mojaloop*, #*design-authority*
5. *Gitter*

ML OSS Community: Design Authority

1. Goals, purpose
2. Membership – driven by contribution
3. Frequency of meetings, boards used
4. Functioning overview
5. Example topics, discussions covered as part of the DA
6. Default owner of the Admin / Operations API and the Settlement API (along with Settlements work stream)
7. DA Board: https://github.com/mojaloop/design-authority/issues#workspaces/da-issue-log-5cdd507422733779191866e9/board?notFullScreen=false&repos=186592307

PI-9 Overview: Focus Areas

1. Performance testing and addressing regression
2. Operational Monitoring - Enhancements
3. Merchant “Request to Pay” – standardization
4. Maintenance: Node, Helm upgrades, Bulk Transfers
5. Testing toolkit – standardizing the PoC



Performance

Testing and Addressing Regression

PI-9: Performance Enhancements

1. Optimized Logging (central-services-logger)
   a) Added conditional logging to reduce unnecessary processing/parsing logic:
      - ML-API-Adapter
      - Central-Ledger
      - Simulator
      - Central-Services-Shared
      - Central-Services-Stream

```
-     Logger.debug('create::payload(%s)', JSON.stringify(request.payload))
+     Logger.isDebugEnabled && Logger.debug('create::payload(%s)', JSON.stringify(request.payload))
```

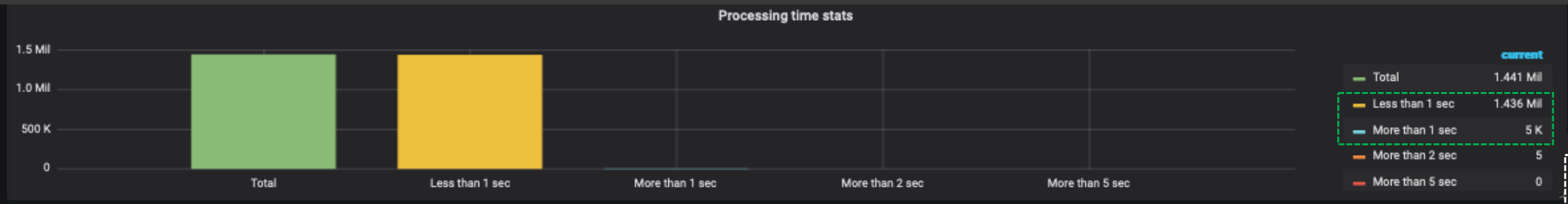
2. Caching of Non-Stateful Data
   a) Participants - central-ledger
   b) Participant Accounts - central-ledger

3. Tracing (event-sdk)
   a) v9.5.x Release supports base64 encoded traceState - containing spanId & custom key-value pairs
      i. timeApiPrepare – unix timestamp for start of Prepare Process
      ii. timeApiFulfil – unix timestamp for start of Fulfil Process

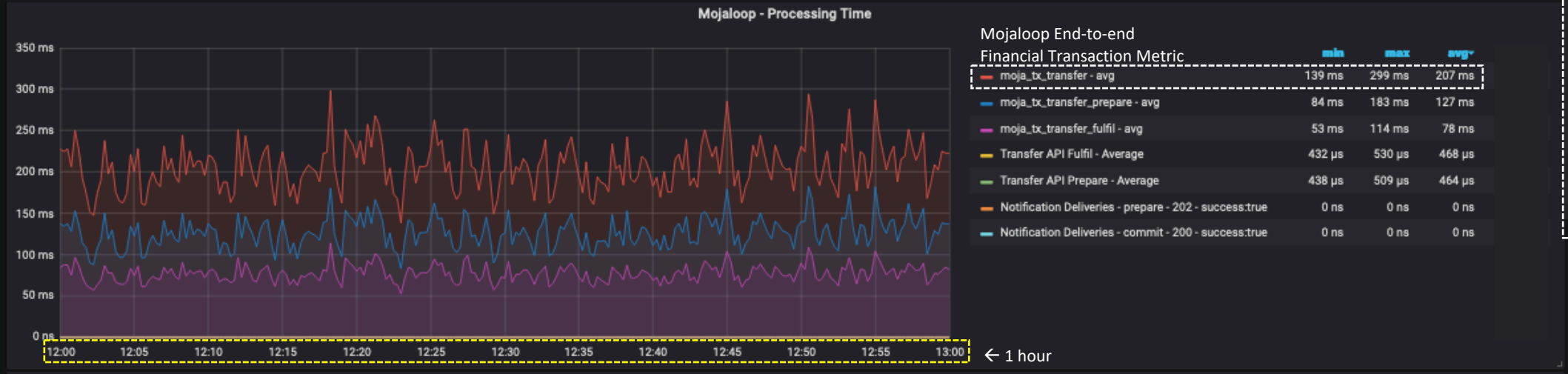
4. Tracing (event-sdk)
   a) v9.5.x Release supports base64 encoded traceState - containing spanId & custom key-value pairs
      i. timeApiPrepare – unix timestamp for start of Prepare Process
      ii. timeApiFulfil – unix timestamp for start of Fulfil Process

5. Metrics (central-services-metrics)
   a) v9.5.x Release supports Histograms, and Summary (new)
   b) New Instrumentations
      i. ML-API-Adapter - reported by Notification Handler
         - Notification Delivery
         - End-to-end Financial Transactions - ‘tx_transfer’
         - Prepare-only Financial Transactions - ‘tx_transfer_prepare’
         - Fulfil-only Financial Transactions - ‘tx_transfer_fulfil’
         - Transaction processing time buckets
         - % of Transaction longer than 1 sec
      ii. Central-Ledger – reported by Prepare, Position & Fulfil Handlers
         - Domain (functional) logic
         - SQL queries, inserts & updates
         - Cache Hits / Misses

PI-9: Performance Baseline - v9.5.x Master Branch



95th Quantile	% > 1 second
353	0.32%

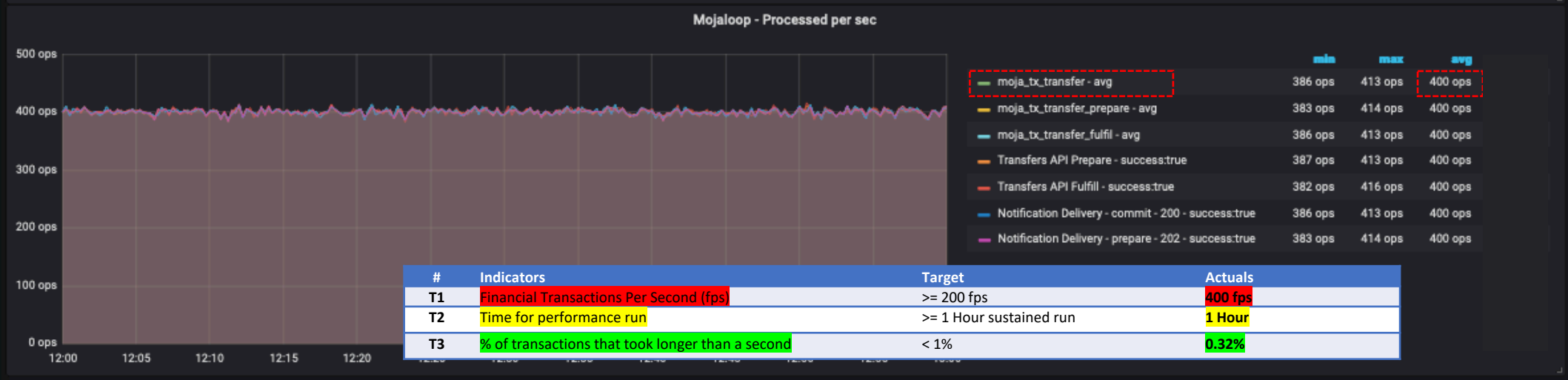


Mojaloop End-to-end Financial Transaction Metric

	min	max	avg
moja_tx_transfer - avg	139 ms	299 ms	207 ms
moja_tx_transfer_prepare - avg	84 ms	183 ms	127 ms
moja_tx_transfer_fulfil - avg	53 ms	114 ms	78 ms
Transfer API Fulfil - Average	432 µs	530 µs	468 µs
Transfer API Prepare - Average	438 µs	509 µs	464 µs
Notification Deliveries - prepare - 202 - success:true	0 ns	0 ns	0 ns
Notification Deliveries - commit - 200 - success:true	0 ns	0 ns	0 ns

Pods / Containers

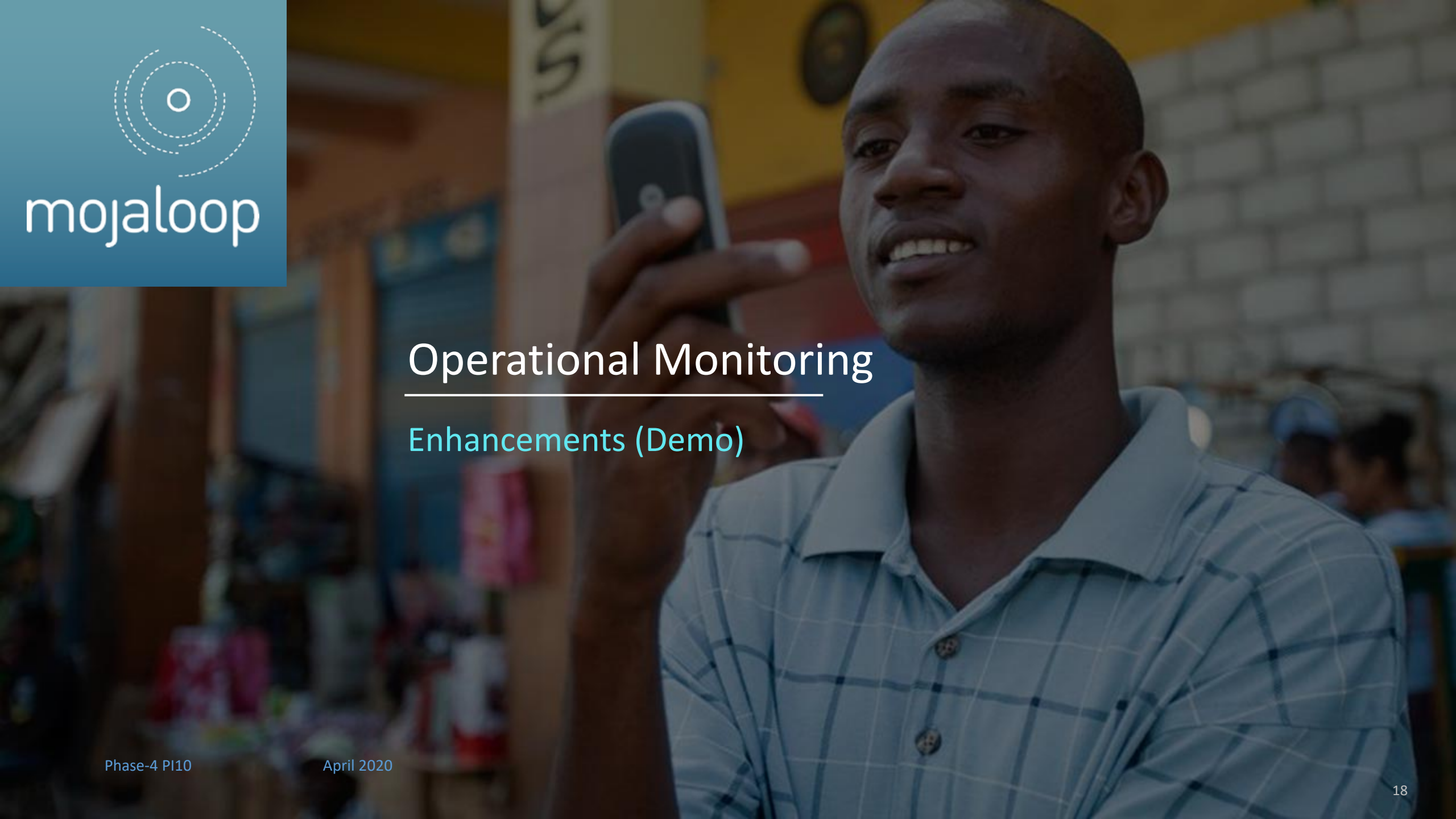
	current
back-centralledger-handler-transfer-position	48
back-centralledger-handler-transfer-prepare	16
back-centralledger-handler-transfer-fulfil	16
back-centralledger-service	1
back-centralledger-handler-transfer-get	0
back-centralledger-handler-timeout	0
back-centralledger-handler-admin-transfer	0
back-ml-api-adapter-handler-notification	32
back-ml-api-adapter-service	8
simulator	32
Total	**153**



	min	max	avg
moja_tx_transfer - avg	386 ops	413 ops	400 ops
moja_tx_transfer_prepare - avg	383 ops	414 ops	400 ops
moja_tx_transfer_fulfil - avg	386 ops	413 ops	400 ops
Transfers API Prepare - success:true	387 ops	413 ops	400 ops
Transfers API Fulfill - success:true	382 ops	416 ops	400 ops
Notification Delivery - commit - 200 - success:true	386 ops	413 ops	400 ops
Notification Delivery - prepare - 202 - success:true	383 ops	414 ops	400 ops

#	Indicators	Target	Actuals
T1	Financial Transactions Per Second (fps)	>= 200 fps	400 fps
T2	Time for performance run	>= 1 Hour sustained run	1 Hour
T3	% of transactions that took longer than a second	< 1%	0.32%

Components	HW Nodes
ML API Adapter – Mojaloop API Service	2
Central-Ledger - Prepare Handlers	4
Central-Ledger - Position Handlers	6
Central-Ledger - Fulfil Handlers	4
ML API Service - Notification Handler	4
Legacy Simulators	4
Database (RDS)	1
Kafka	5
Zookeeper	
Total	**30**



Operational Monitoring

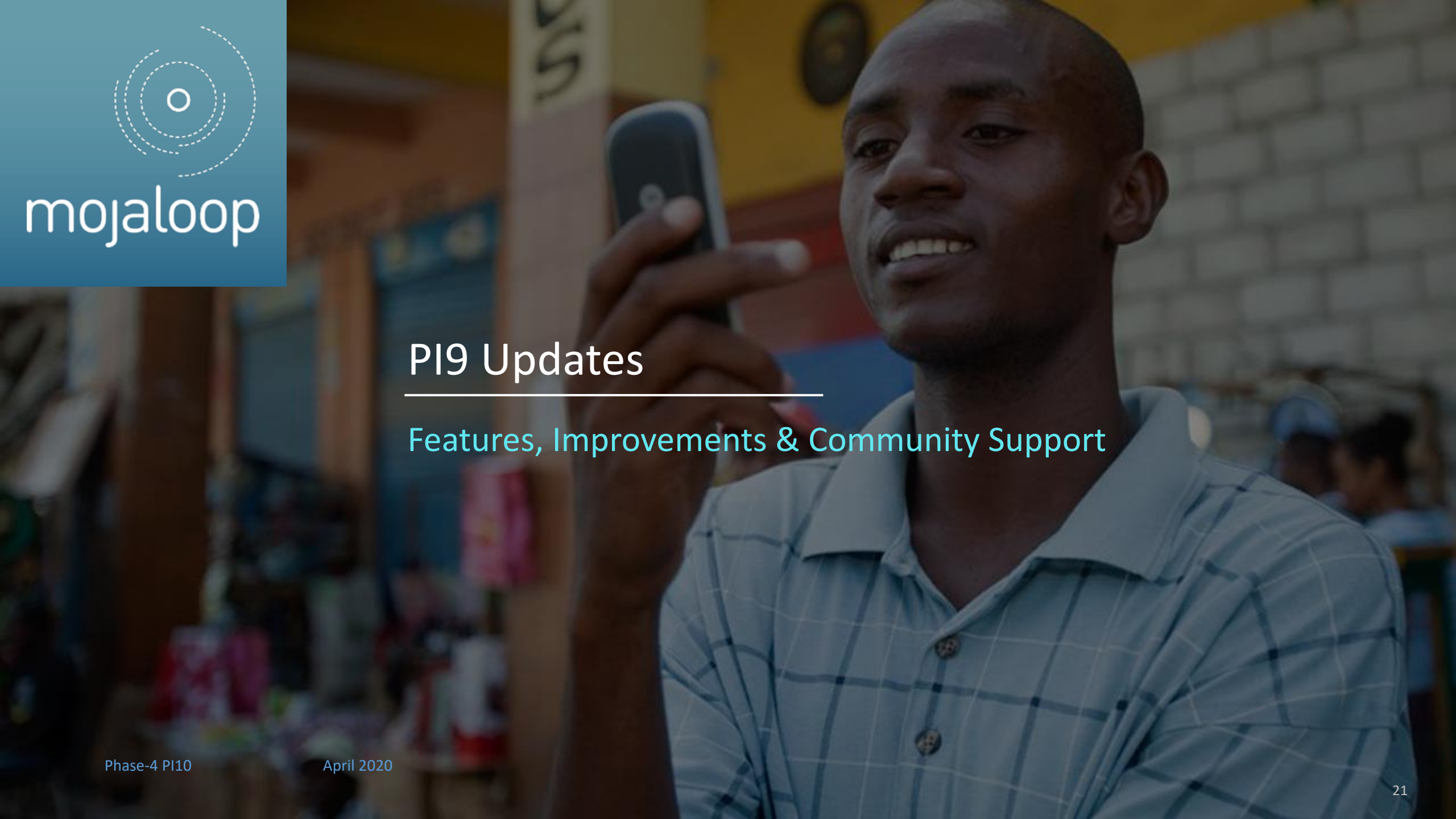
Enhancements (Demo)

PI-9: Operational Monitoring - Demo

1. Grafana Dashboards updated to display the following:
   a. ML-API-Adapter - reported by Notification Handler
      - Notification Delivery
      - End-to-end Financial Transactions - ‘tx_transfer’
      - Prepare-only Financial Transactions - ‘tx_transfer_prepare’
      - Fulfil-only Financial Transactions - ‘tx_transfer_fulfil’
      - Transaction processing time buckets
      - % of Transaction longer than 1 sec
   b. Central-Ledger – reported by Prepare, Position & Fulfil Handlers
      - Domain (functional) logic
      - SQL queries, inserts & updates
      - Cache Hits / Misses

PI-9: Transaction Requests Service

1. Transactions requests service
    a. Metrics added
    b. Event Framework added
    c. Error Framework added
    d. Postman tests and QA
2. End-to-end support for Merchant ‘ Request to Pay’ Use Case



PI9 Updates

Features, Improvements & Community Support

PI-9: Maintenance

1. Node Upgrade – LTS 12.16.x
    a. Improved security
    b. Improved performance
    c. Code quality and ease of maintenance
2. Helm v3 Support
    a. Current charts support v3 and v2
    b. Backwards compatibility of charts with v2 verified on ML OSS environments
    c. Helm v3 is easily maintainable and more secure (No server-side changes needed; Tiller not needed)

PI-9: Bulk Transfers

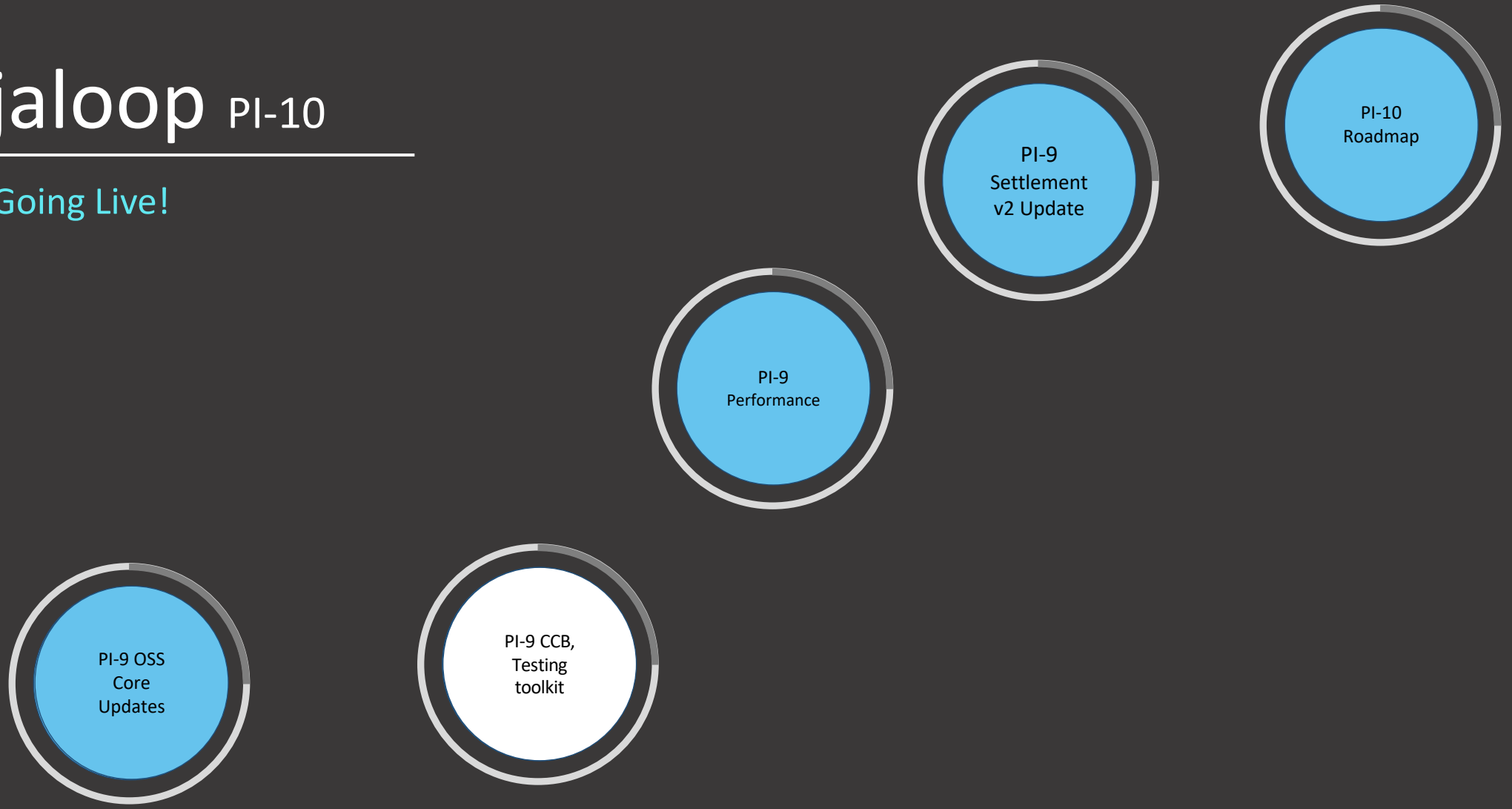
1. Bulk Transfers
    a. *PoC done in PI7*
    b. PI9: Happy Path standardized with tests and Simulator support
    c. PI9: Postman tests and QA
2. Needs to be prioritized for PI10 (Roadmap) to provide complete functionality

PI9: Additional Updates

1. Accents in names
    a. Allowing names with accent characters in names
    b. Solution in progress based on Regular Expression for nodejs
2. Helm Releases, validation and maintenance
3. Bug fixes – 18 bugs fixed (documented as issues)
4. Community Support: Slack support regarding Specification, implementation clarifications

Mojaloop PI-10

Phase4 Going Live!

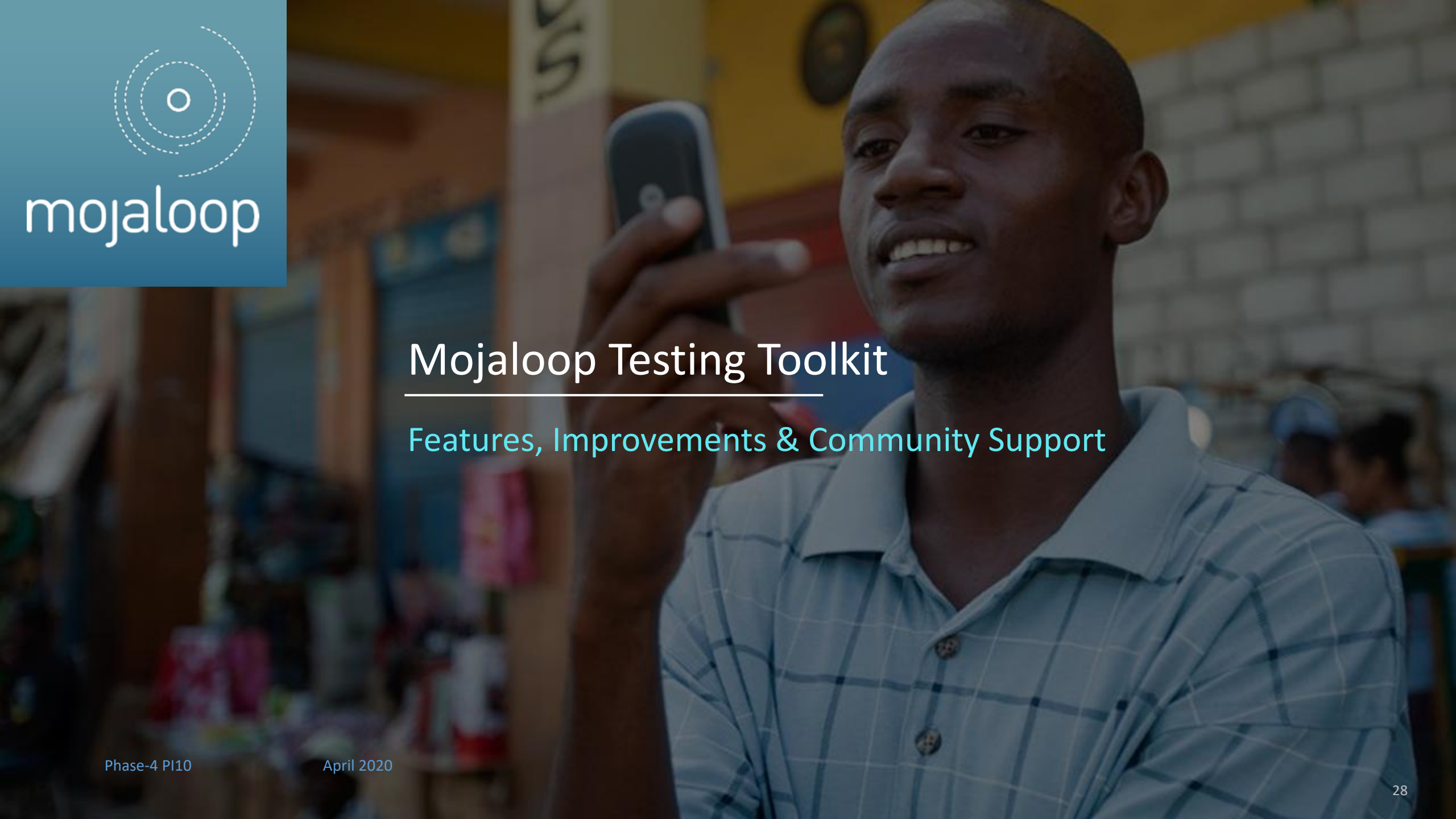


ML OSS Community: Change Control Board

1. Goals, purpose – ML FSPIOP API
2. Current Membership
    i. BMGF (Matt Bohan, Miller Abel)
    ii. Ericsson (Henrik Karlsson)
    iii. Huawei (Chen Hill)
    iv. Mahindra Comviva (Ritvik Sinha)
    v. ModusBox [Non-voting] (Michael Richards, Sam Kummary)
    vi. Mowali (John Mark Ssebunya)
    vii. Telepin (RJ Wilson)
    viii. BoT TIPS (Mutashobya Mushumbusi)
3. Frequency of meetings, boards used
4. Change requests, Solution proposals, Bugs

CCB – ML FSPIOP API: Publishing v1.1

1. Current status:
    a. Drafts available for public review
2. Overview of changes:
    a. List of changes: https://github.com/mojaloop/mojaloop-specification/issues/52
    b. Option to notify completion of a transfer to a Payee
    c. Updating description, examples to address inaccuracies and omissions
    d. Clarify usage of ABORTED state in the Fulfil step of a transfer
    e. Describe quote rejection flow
    f. Add notes to indicate the case-insensitive nature of HTTP headers
    g. Clarify usage of FSPIOP-Destination header
    h. Adding ExtensionList element to several items in the data model

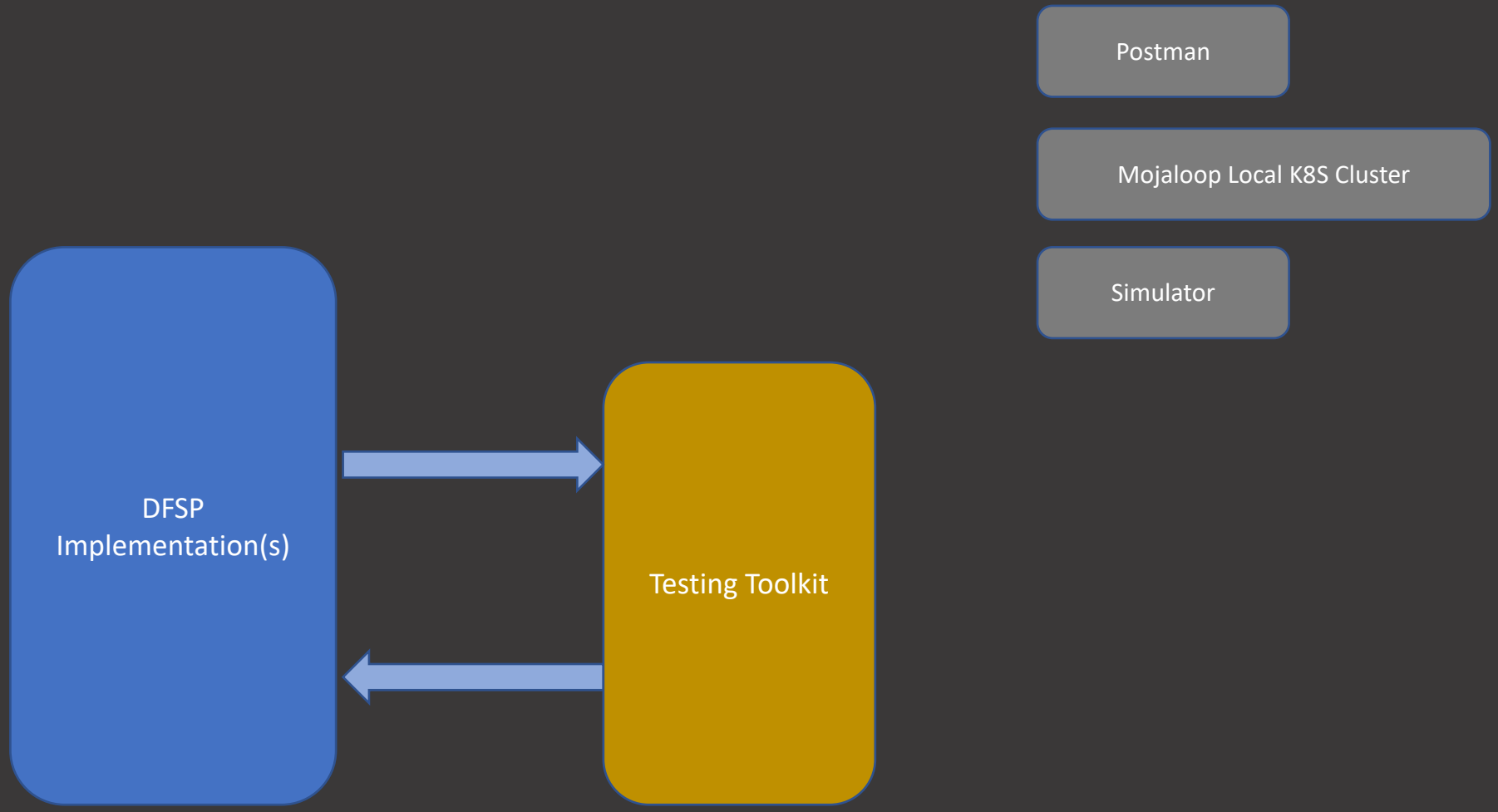


Mojaloop Testing Toolkit

Features, Improvements & Community Support

Phase-4 PI10 April 2020

What is the Testing Toolkit



Testing toolkit: Goals

1. Test the Mojaloop API implementations
2. Easy & simple to use
3. Support different versions of Mojaloop API & other APIs
4. Highly configurable (With Rules Engine)
5. Can validate Inbound requests
6. Can generate Outbound requests and assert on responses and callbacks

Toolkit Features

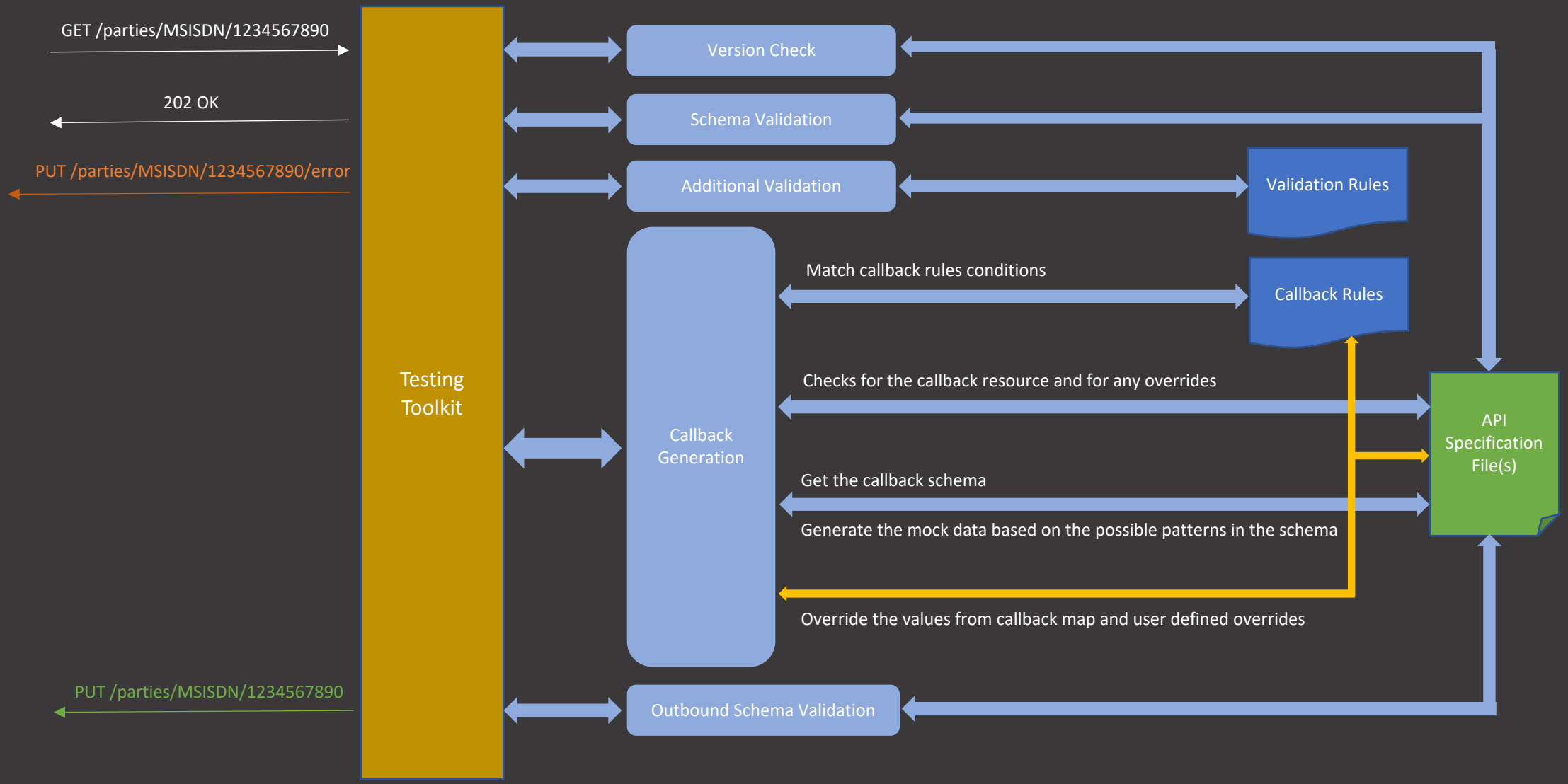
1. User Interface for QA / Product / Business users
2. Version validation and negotiation
3. Schema validation
4. Additional validation and error callback generation
5. Dynamic callback generation based on rules
6. Initiation of use cases (outbound)
7. Simultaneous support for multiple APIs
8. Synchronous & Asynchronous APIs
9. Assertions to validate desired behavior
10. Supports JWS and mTLS

What's new in PI9

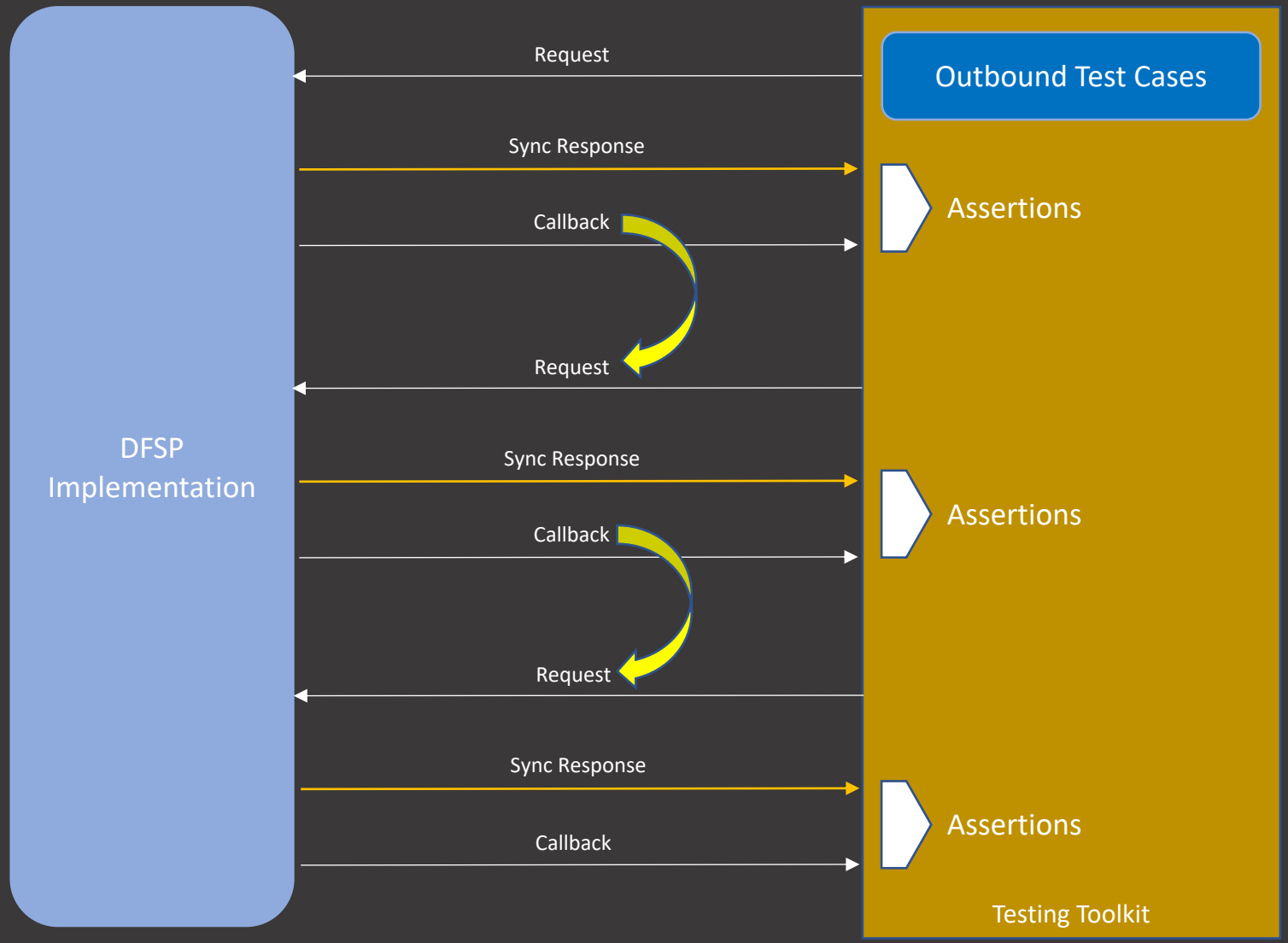
1. Support for multiple APIs simultaneously (Including sync & async APIs)
2. Validate association of a transfer request with an existing quote
3. Settings page in front end
4. Automated tests (CI / CD) for git repositories
5. Test coverage
6. Added settlements API
7. Assertions on test cases
8. JWS & mTLS using connection manager

Testing toolkit: Demo

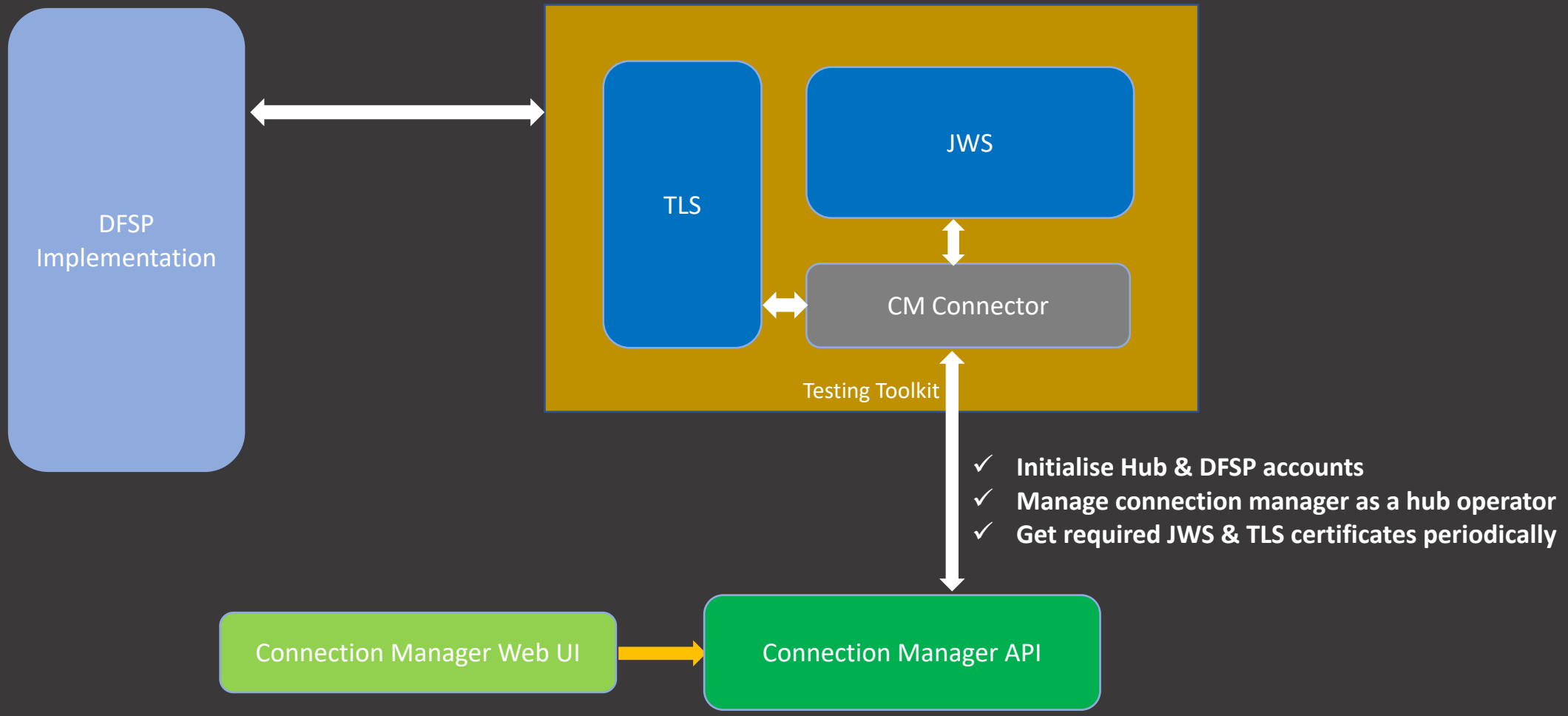
How Testing Toolkit Works - Incoming requests



How Testing Toolkit Works - Test Case Initiation and Assertions



Security - JWS & TLS

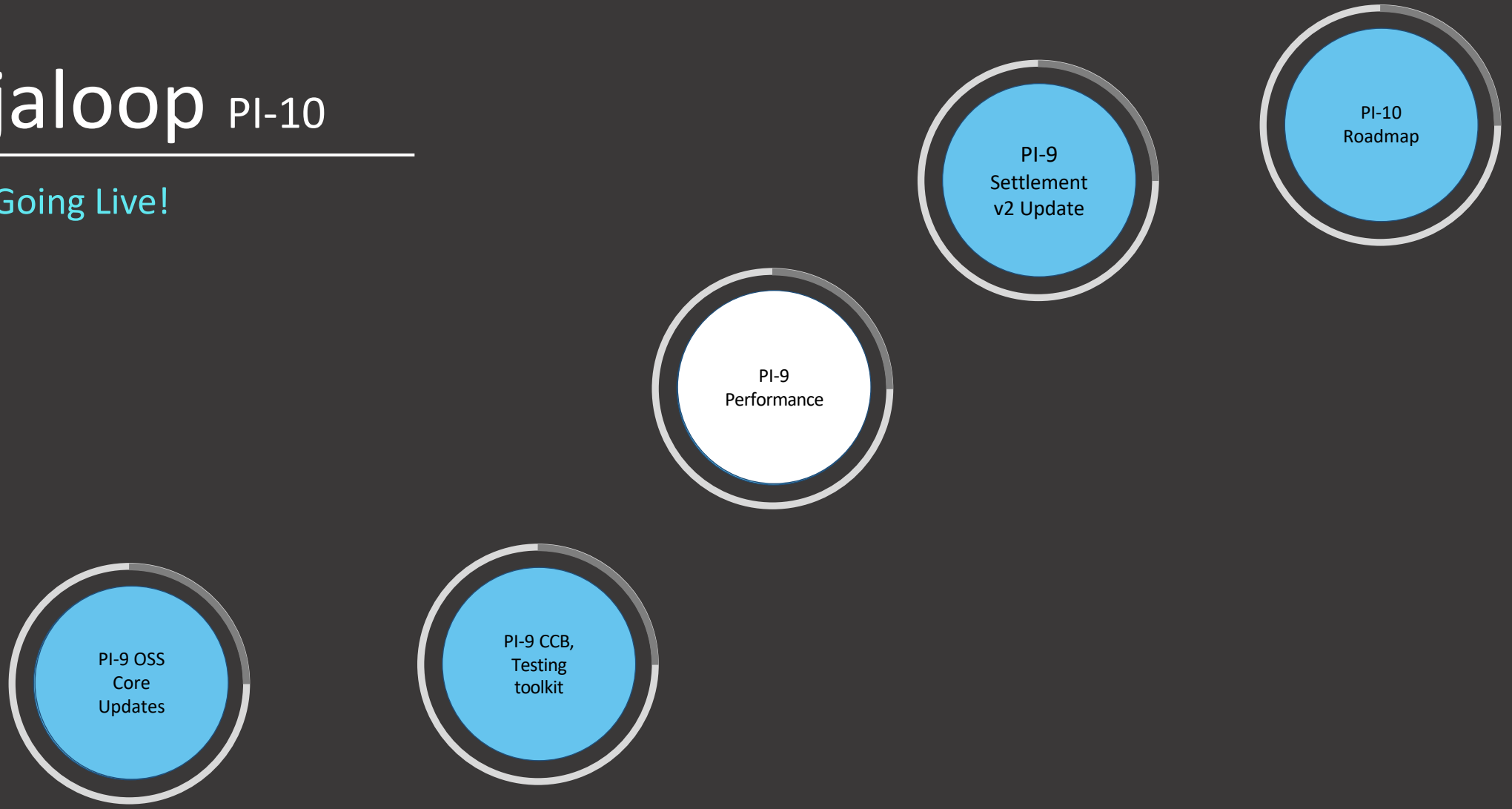


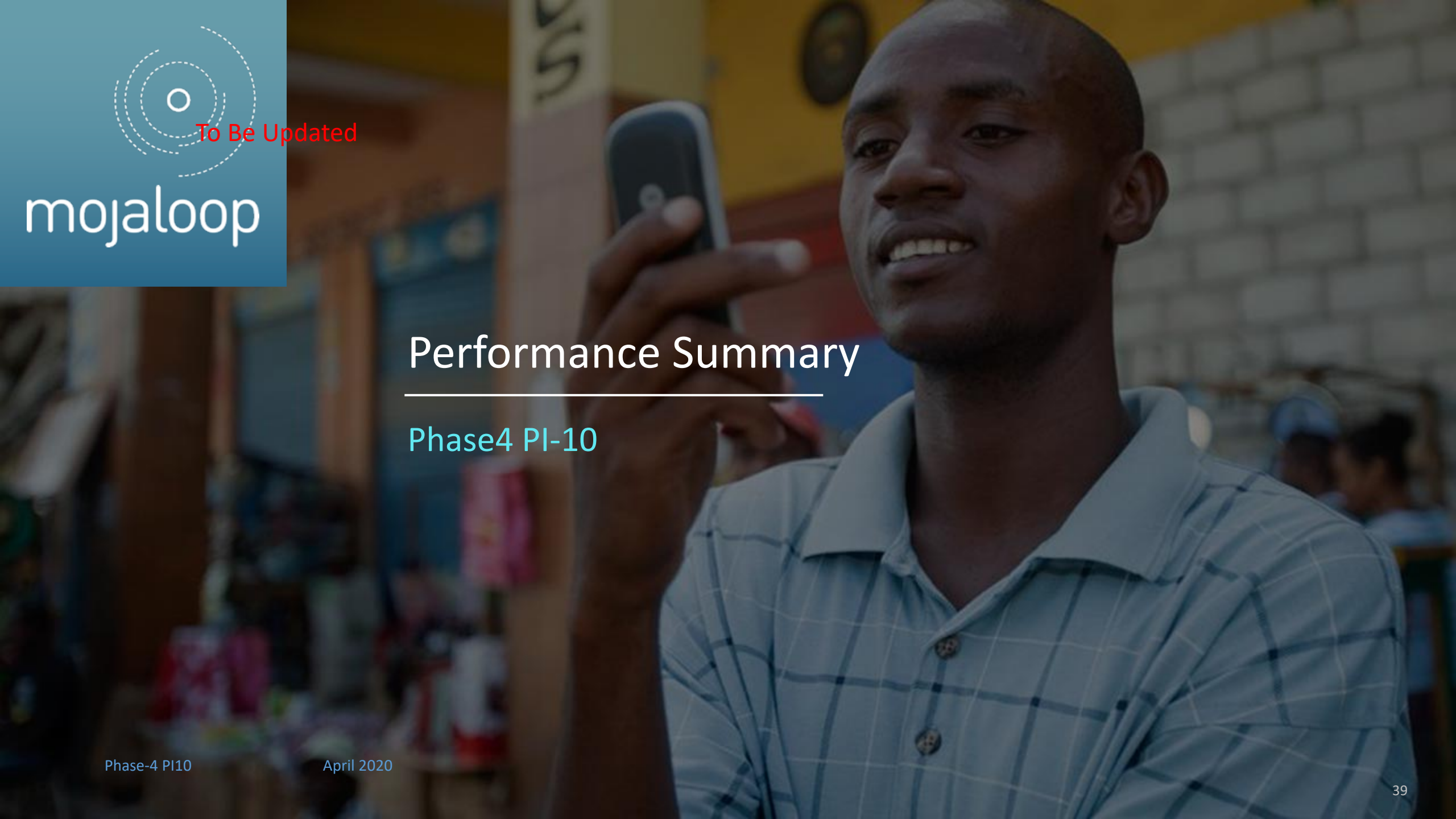
Testing toolkit: Roadmap

1. Prioritize rules
2. Import and Export capability of rules and settings as files
3. Expand support for all the resources in API spec file
4. Command line tool for using testing toolkit in devops automation systems
5. QA – Automated UI end to end tests using testcafe or cypress & > 98% test coverage
6. Support for hosting the testing toolkit
7. Performance Tuning to withstand load (not for perf testing)
8. Event framework
9. Cache
10. Dashboard
11. Incorporate prioritized items from feedback, community input

Mojaloop PI-10

Phase4 Going Live!





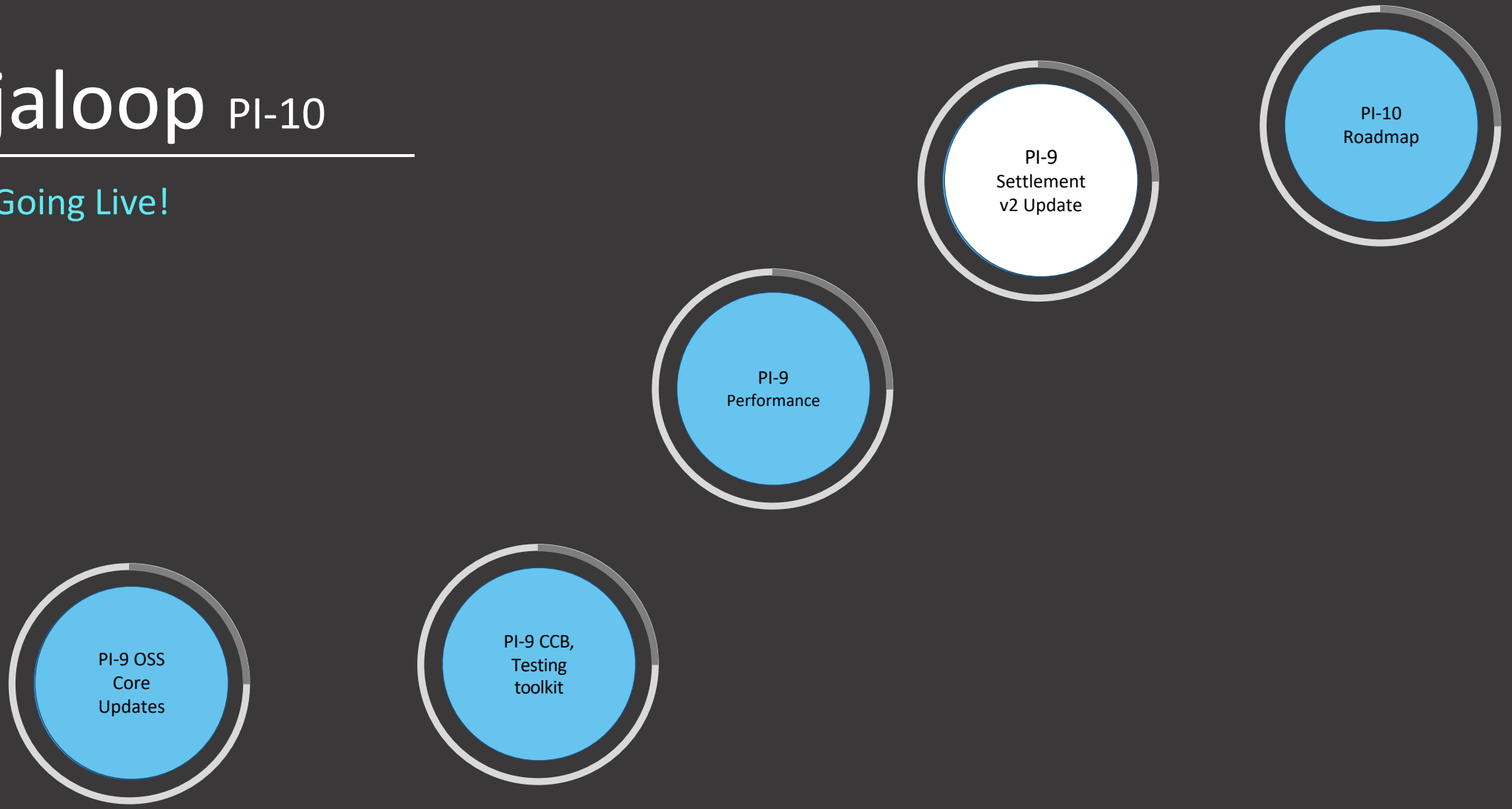
To Be Updated

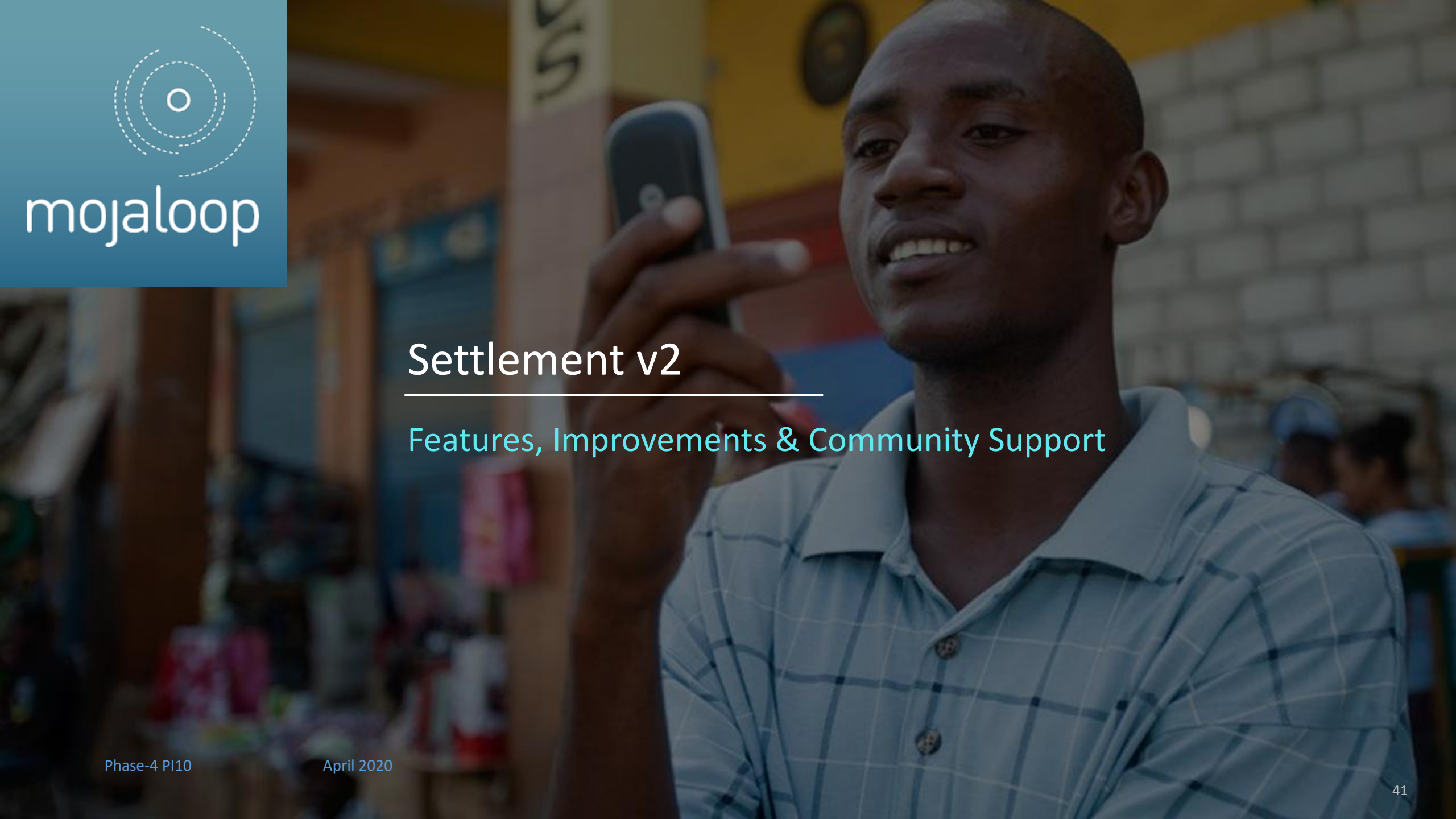
Performance Summary

Phase4 PI-10

Mojaloop PI-10

Phase4 Going Live!





Settlement v2

Features, Improvements & Community Support

Defining scheme settlement models

Parameter	Description
Name	A user readable name for the model e.g. ContinuousGrossPosition, DeferredNetInterchangeFee
Granularity	• GROSS: Per transfer • NET: Aggregate of transfers
Interchange	• BILATERAL: Participants settle with each other • MULTILATERAL: Participants settle with the scheme. Each participant gets a bill for the NET value of their transfers.
Delay	• IMMEDIATE: Actioned immediately after completion • DEFERRED: Actioned after a period of time
Currency	• Currency Code: Only includes a specific currency • NULL: Creates an aggregate by currency code
Ledger Account Type	Principal values and interchange fees can be settled in different ways i.e. POSITION, INTERCHANGE_FEE, other account types can be created.
Requires Liquidity Check	Whether the settlement model requires NET_DEBIT_CAP to be checked.
Adjust Position	Whether the participant's POSITION is automatically adjusted when settlements are completed.

OSS Settlement v2 – Goals

1. Supporting different settlement models within the same scheme
2. Multi-lateral Net Settlement (MLNS) models
3. Support models that don’t support liquidity checks e.g. Interchange fees
4. Enhancements to support Continuous Gross Settlement

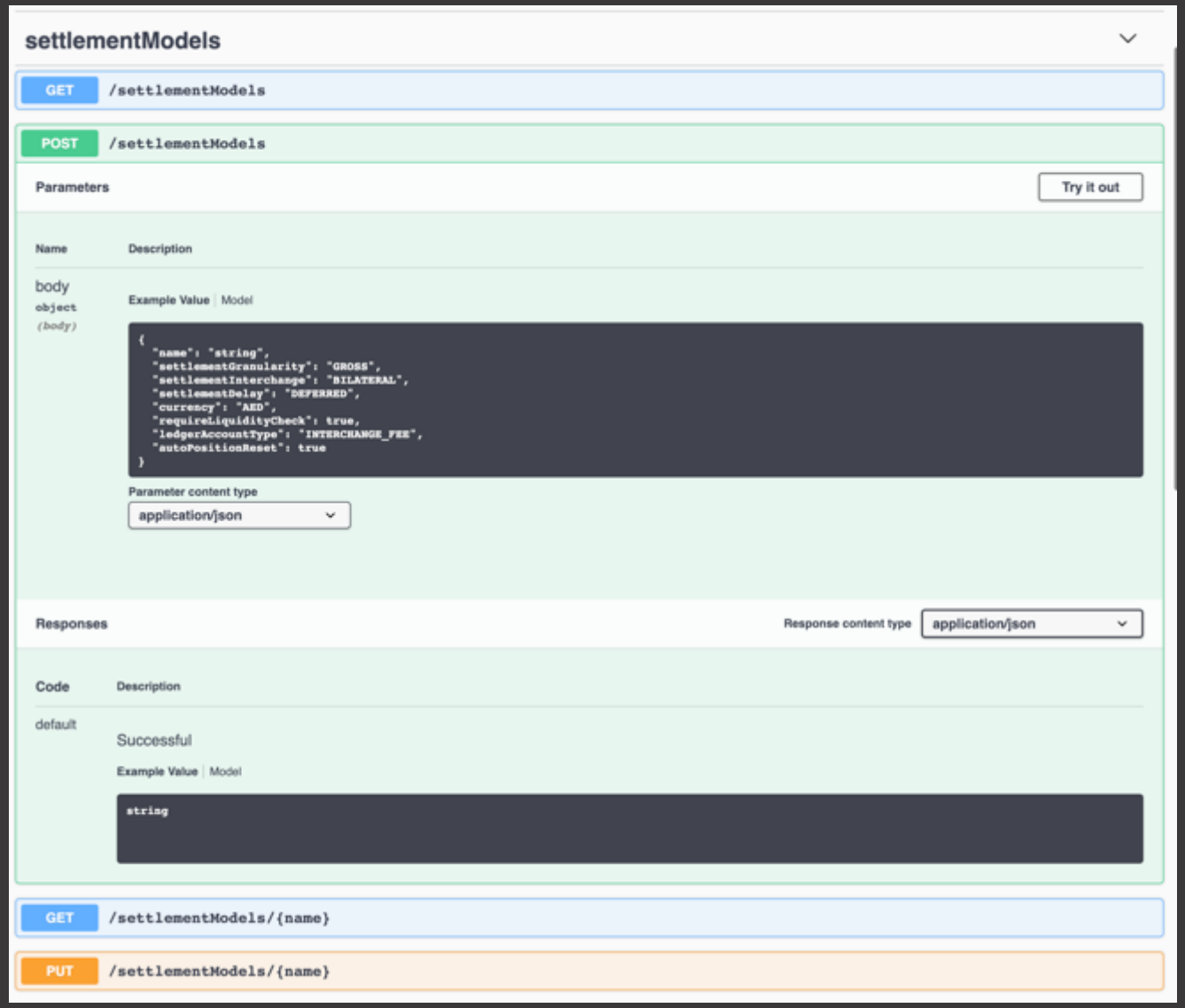
OSS Settlement FSD Outline

Available at Mojaloop Documentation

- https://mojaloop.io/documentation/mojaloop-technical-overview/central-settlements/oss-settlement-fsd.html

1. Definition of terms: ledger account entry types, states, models
2. Business rules: how it is working currently? Key facts
3. Settlement V2 enhancements:
    1. Request settlement by currency and account type
    2. Support continuous gross settlement
    3. Processing ledger account types (e.g. TIPS Interchange fees)
    4. Automatic position reset (to support Mowali)
        1. Add “adjustPosition” column on settlement model table defaulted to false
        2. Implement logic in position handler.

Central Ledger API – Settlement Models (v9.5.0)



Settlement Window Handling

1. Key Objective: No additional overhead to be imposed on transfer throughput.
2. Current synchronous process for closing a settlement window: OPEN -> CLOSED.
3. In order to support Gross settlement models we need to support status per transfer and not per aggregation as is presently done for Net.
4. Generation of window content and aggregations by ledger account type and currency includes transfer status.
5. Asynchronous processing and retries.
6. Settlement window state management and content list.

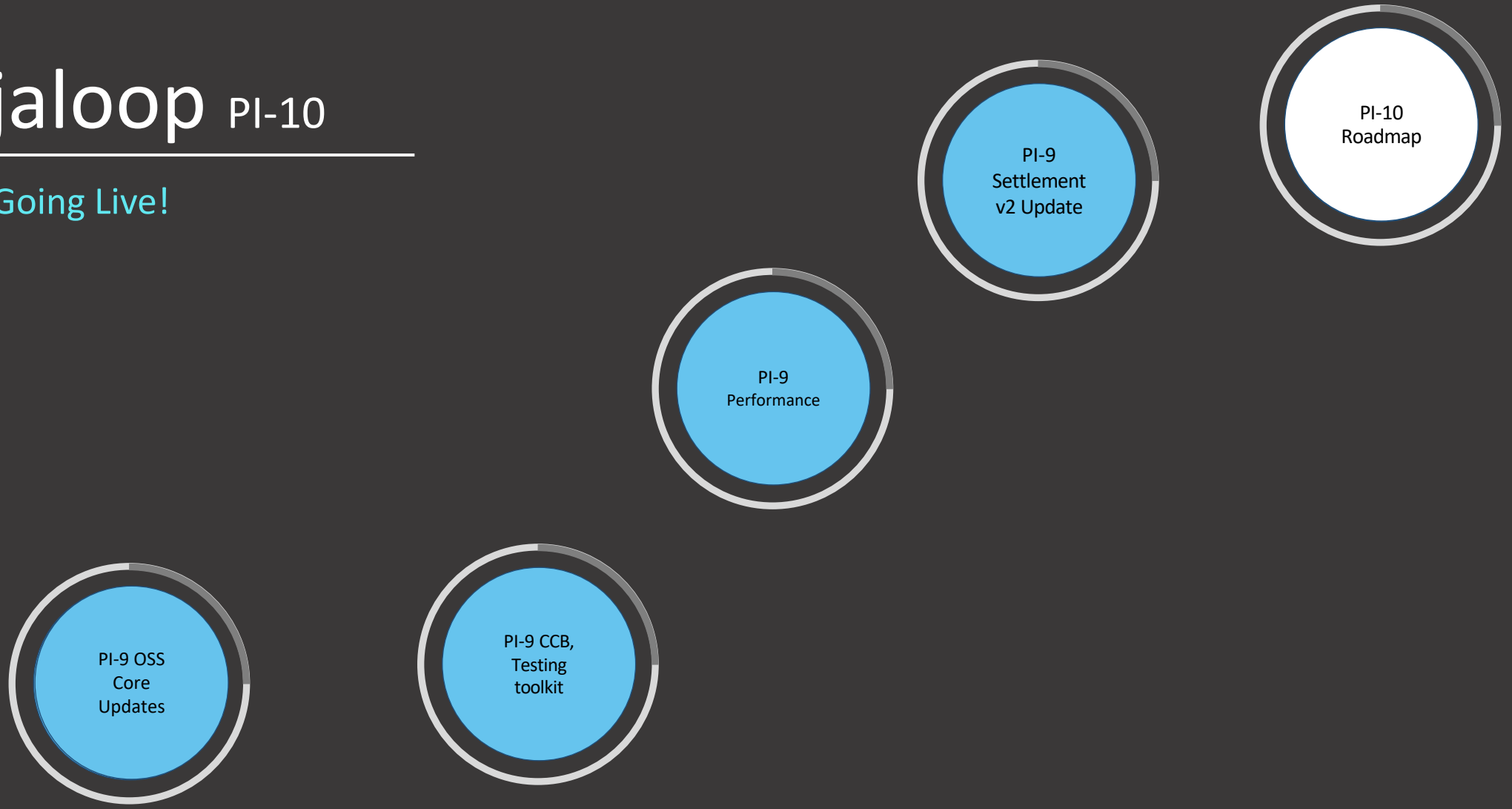
Settlement v2 Roadmap

1. Settlement by currency and account type
   a. Completed end to end testing with current Golden Path
   b. Enhance Golden Path with additional currencies and multiple windows
2. Continuous gross settlement
3. Processing of interchange fees to support TIPS
4. Settlement transfer and automatic reset of positions
   a. Add “adjustPosition” to settlement model table
   b. Implement automatic position reset functionality
5. Updating Finance Portal to enhance and align settlement functionality
6. V2 migration strategy

Settlement v2: Golden Path Walkthrough

Mojaloop PI-10

Phase4 Going Live!



Phase-4: Roadmap - Ongoing

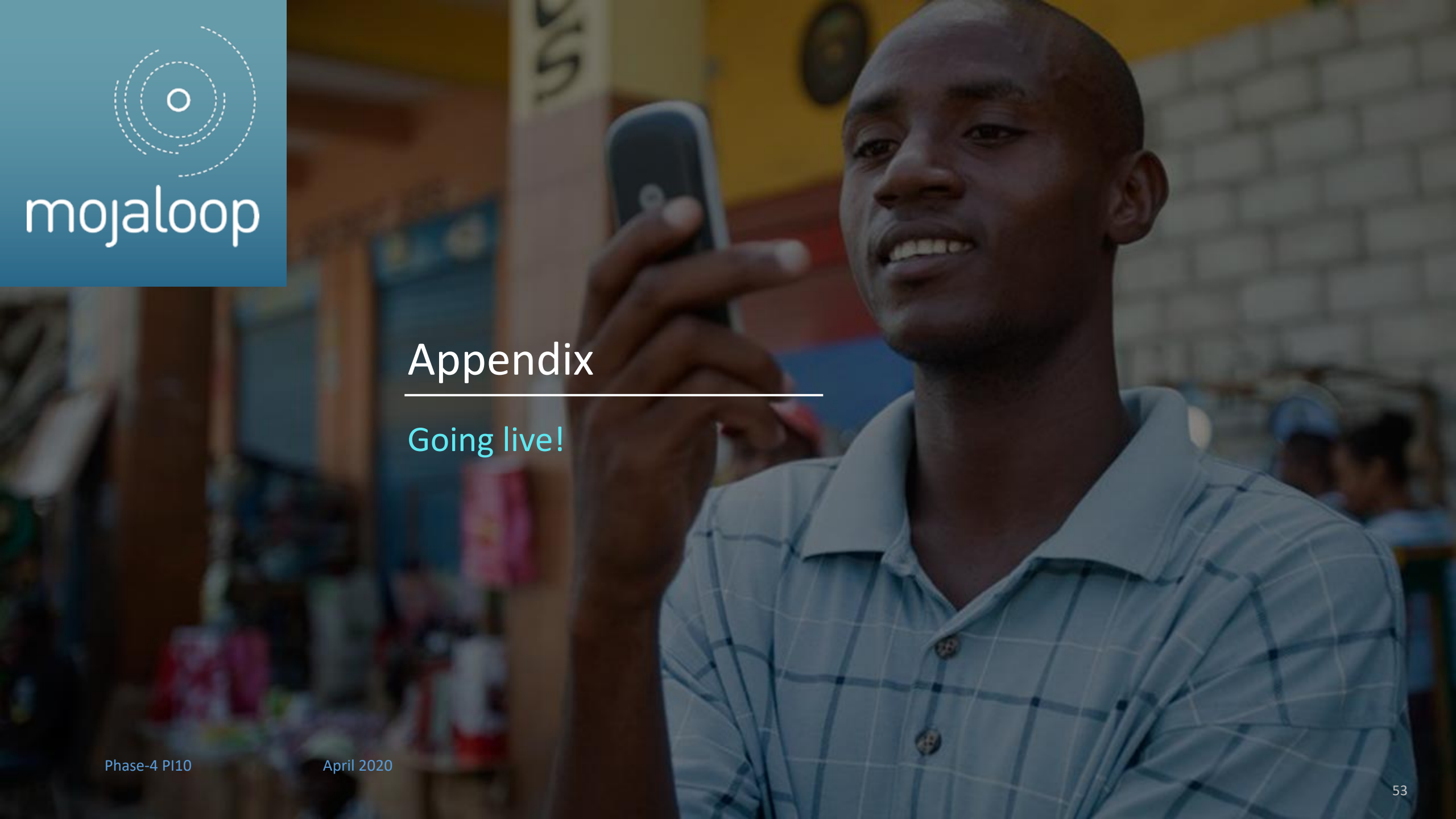
1. Settlement v2
2. Versioning standards, version maintenance
3. Cross Network / Currency (CNP/FXP)
4. Code quality and improvements
5. Performance Testing, Improvements of the Mojaloop Core
6. Fraud Management
7. Leadership & Community Management (events, governance, communication, tools, etc.)
8. Payment Initiation Service Provider – Solution
9. LPS Adapter Enhancements and future Use Cases (ATM, POS)
10. Testing tool-kit

Phase-4: Roadmap – To be prioritized

9. Bulk transfers implementation, QA
10. Standardize operations (Admin) API
11. Portal for Hub Operations and Hub TechOps
12. Secure auditing / Forensic Logging
13. PoC for a ticketing system
14. Rules API
15. API Gateway
16. Split Payment Capability (SNAPP)
17. Payment Hub Integration (MIFOS)
18. Payments Gateway

Phase-4: Roadmap – Legend

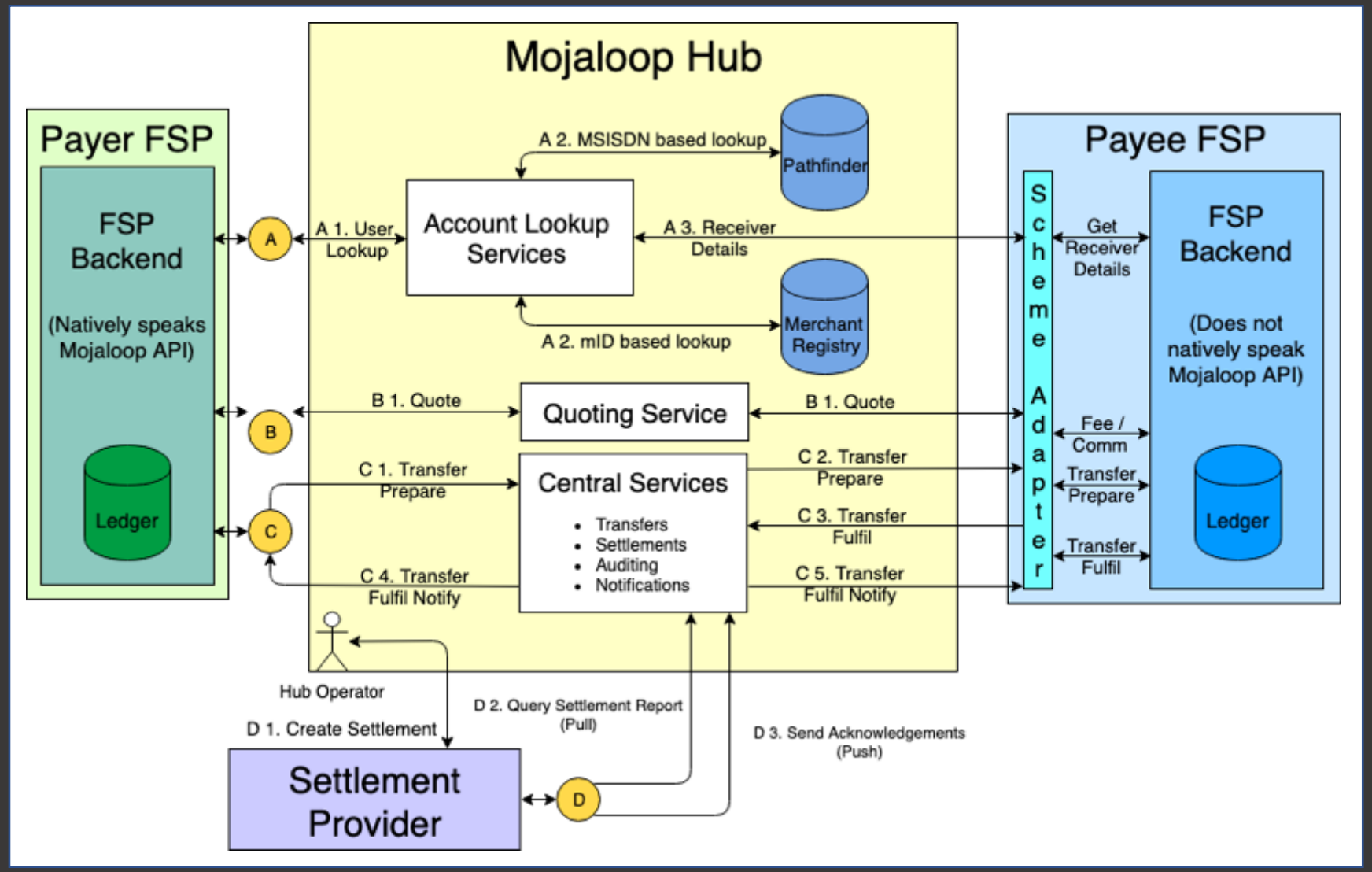
1. Hygiene = Code standards + QA + Code standards
2. Known interested Commercial implementers
    1 - Mowali
    2 - TIPS
    3 - Kifiya
    4 - Google
    5 - Paysys
    6 - Mifos



Appendix

Going live!

Mojaloop Overview



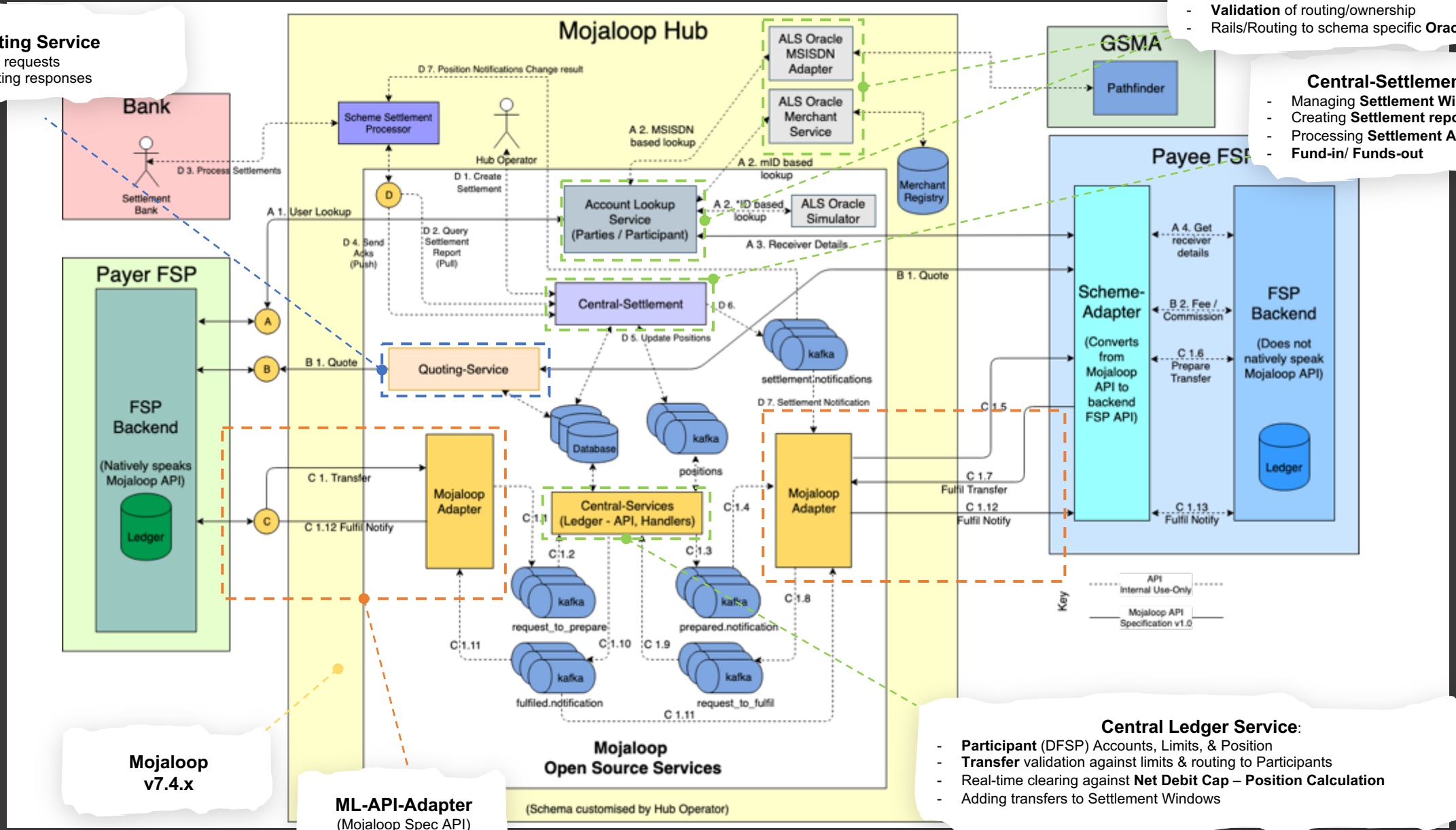
DEMO: http://mojaloop.io/docs/CentralServices/mojaloop-architecture-static-demo/index.html

High-level Architecture (PI7)

Account Lookup Service:
- Resolve **Participant** (DFSP) Routing
- Resolve **Party** (Customer) details via DFSP
- **Validation** of routing/ownership
- Rails/Routing to schema specific **Oracle registries**

Central-Settlement Service:
- Managing **Settlement Windows**
- Creating **Settlement reports**
- Processing **Settlement Acks**
- **Fund-in/ Funds-out**

Quoting Service
- Initiate Quote requests
- Resolve Quoting responses

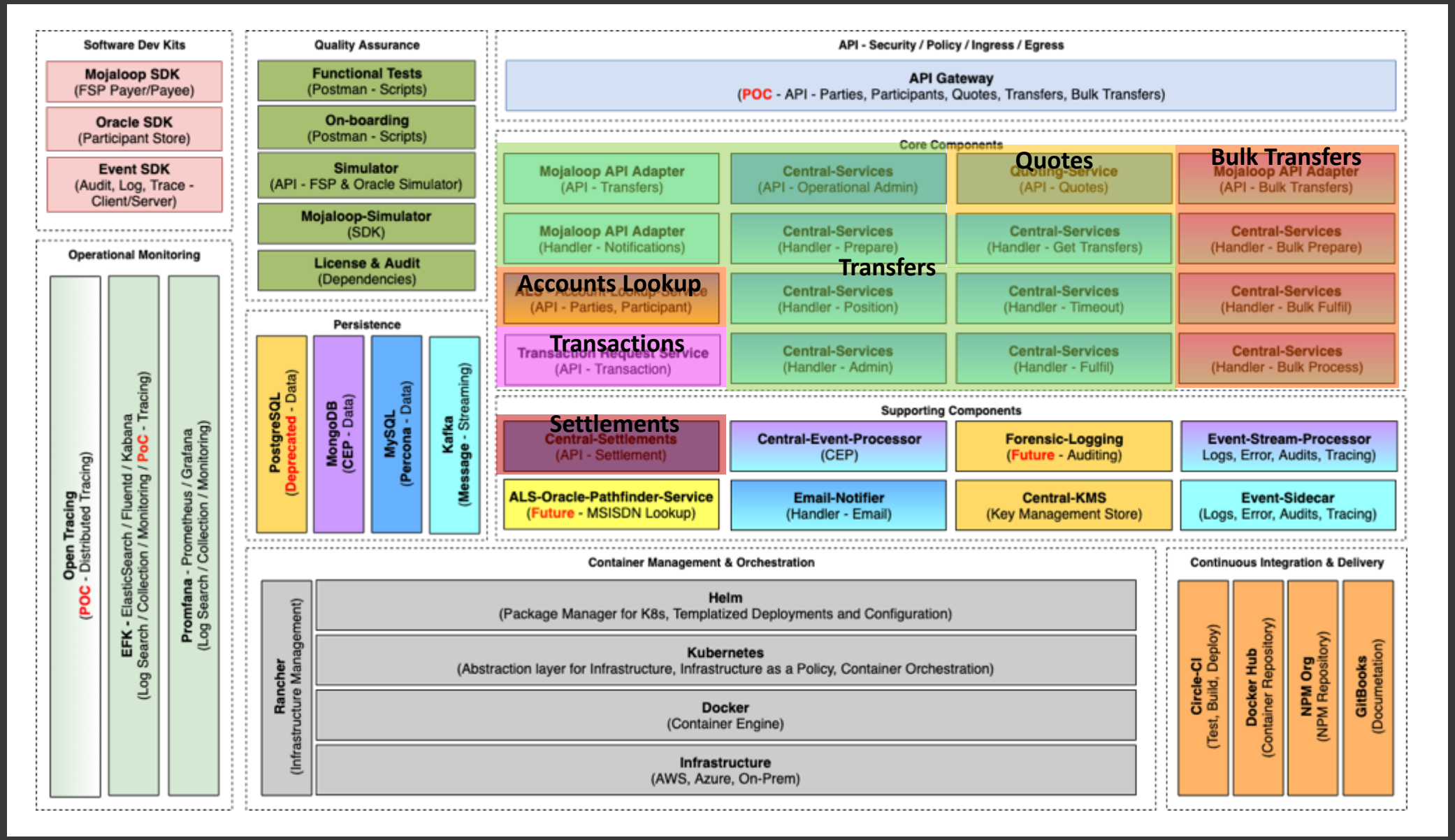


Mojaloop v7.4.x

ML-API-Adapter
(Mojaloop Spec API)

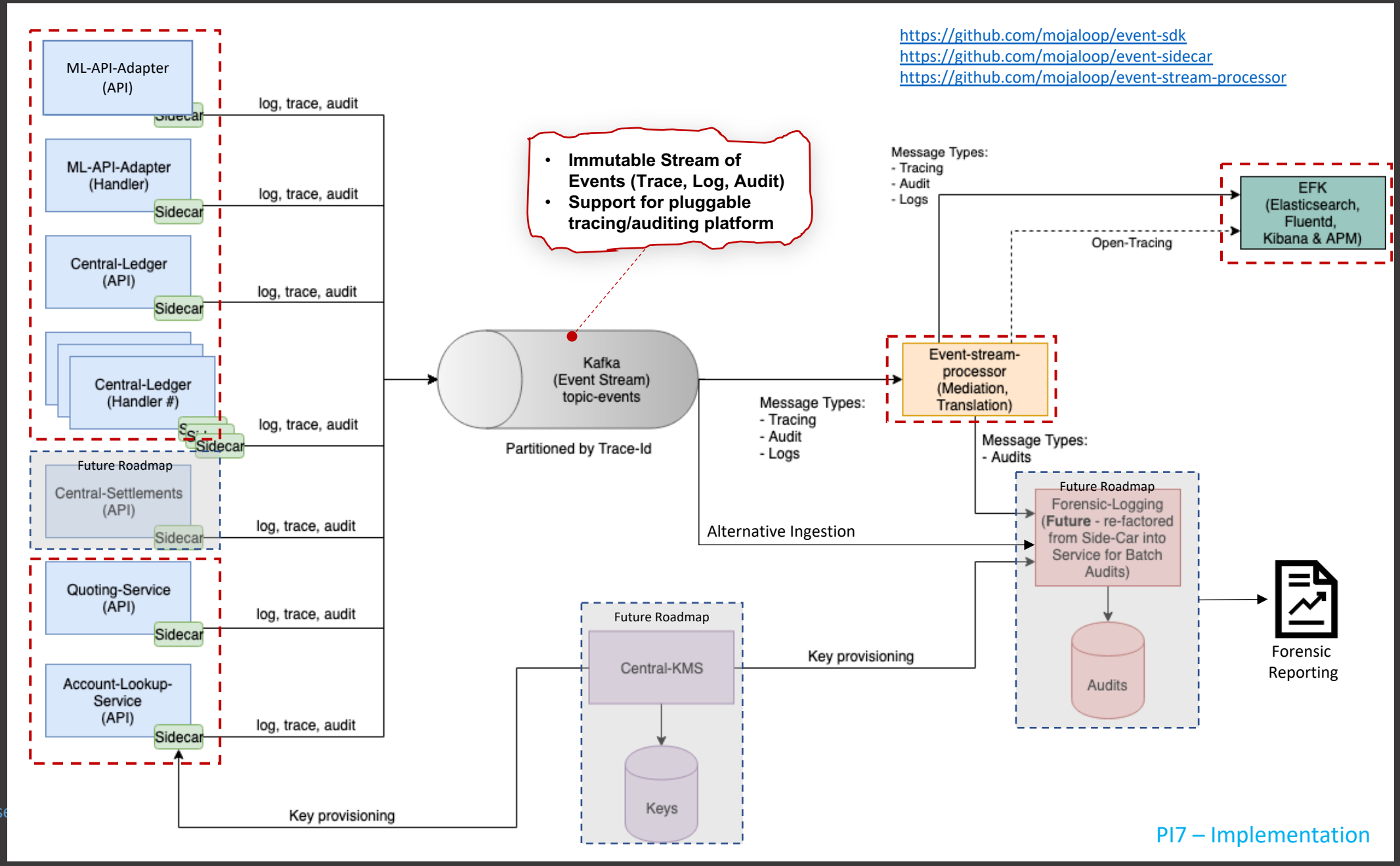
Central Ledger Service:
- **Participant** (DFSP) Accounts, Limits, & Position
- **Transfer** validation against limits & routing to Participants
- Real-time clearing against **Net Debit Cap** – **Position Calculation**
- Adding transfers to Settlement Windows

Component Overview (PI7)



* Color-shading of core & supporting components match dependent persistent stores.

Event Framework – Current Functional Overview



Run-time Platform – Kubernetes

What is Kubernetes?
Open-source system for automating deployment, scaling, and management of containerized applications.

kubernetes

Why Kubernetes?

Deploy your applications quickly and predictably
- Infrastructure as a Policy
- Abstraction of Infrastructure (Cloud, On-Prem)

Scale your applications on the fly
- Policy rule based scaling
- Elastic scaling (horizontally up/down)
- Limit hardware & resources via Policies

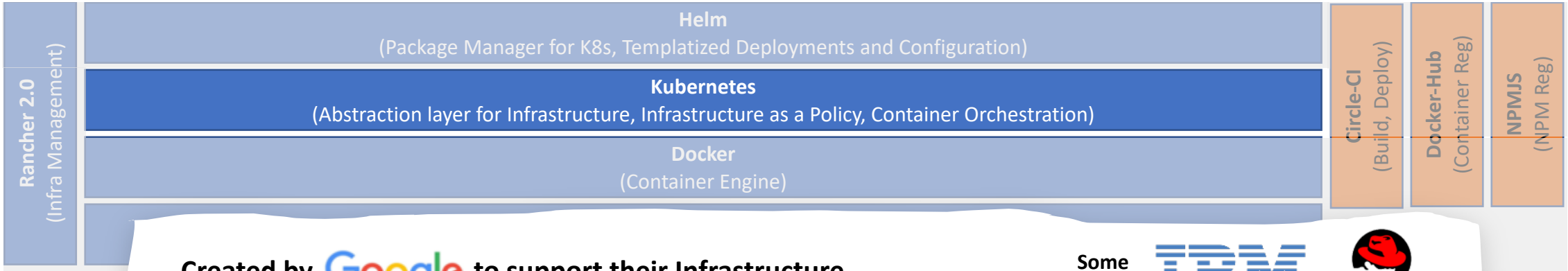
Roll out new features seamlessly
- Rolling updates

Discoverability
- Dynamic service resolution via DNS

Durability
- Self-healing
- Auto-[placement, restart, replication, scaling] based on Policies
- Load Balancing

Security
- Isolation through Containers, Network and Namespaces

Operations
- App config & secrets stored in distributed key-value store (etcd)
- Monitoring of containers



Created by Google to support their Infrastructure.

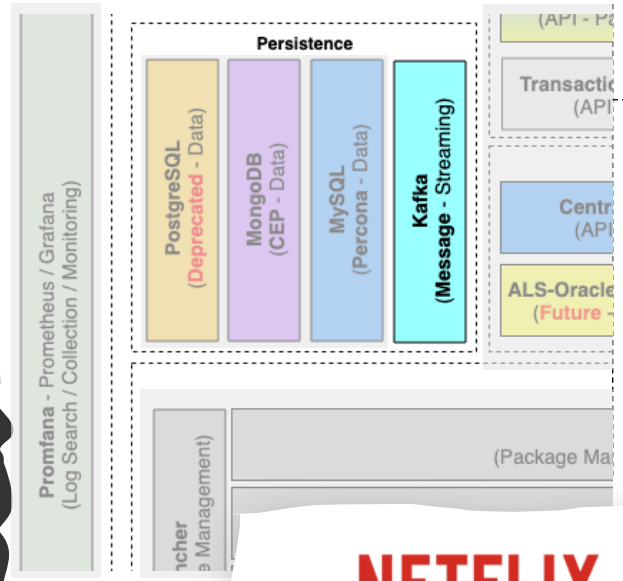
Some contributors: IBM redhat

Messaging Platform – Kafka

Ref: https://kafka.apache.org/

What is Kafka?

Apache Kafka is a distributed message streaming platform.

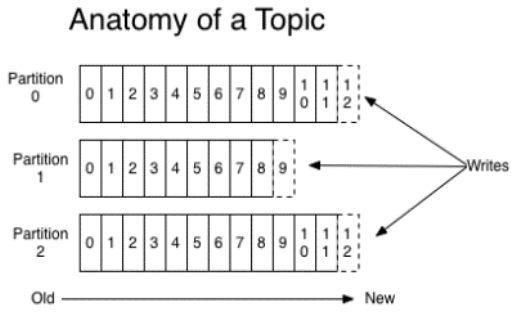


Why Kafka?

Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.

Store streams of records in a fault-tolerant durable way with history being saved for a desired period.

Process streams of records as they occur.



- For each topic, the Kafka cluster maintains a partitioned log.
- Each partition contains a sequence of records that is
  - Ordered; and
  - Immutable
- The records in the partitions are each assigned a sequential id number called the *offset* that uniquely identifies each record within the partition.

Used By

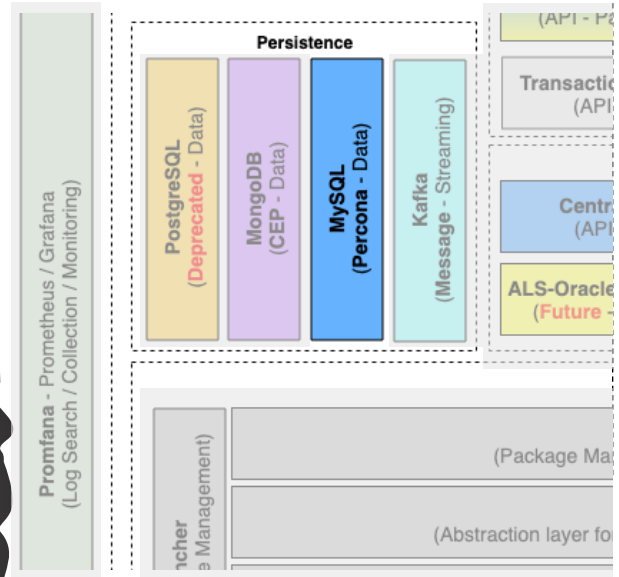
NETFLIX PayPal ORACLE IBM Rabobank

Storage Platform – Percona XtraDB Cluster

PERCONA

What is Percona XtraDB?

An open source, cost-effective, and robust MySQL clustering solution for businesses.



Ref: https://www.percona.com/software/mysql-database/percona-xtradb-cluster

Why Percona XtraDB?

- Cost-effective HA and scalability for MySQL with both Open Source and Enterprise support options
- Increased read/write scalability
- Zero data Loss
- Multi-master replication
- Works on-premises, cloud, hybrid, WAN, LAN, Kubernetes support (Helm)

Used By

Deployment Architecture – Rancher Overview

Ref: http://rancher.com

What is Rancher?

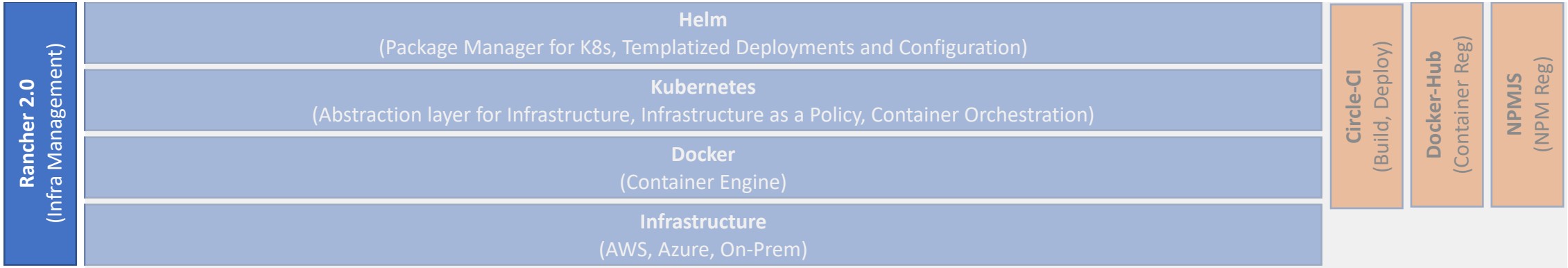
Rancher is enterprise management for Kubernetes.

Every distro. Every cluster. Every cloud.

Why Rancher?

- Kubernetes Management (v1.8 to *v1.9)
- Container Management
- *Access Management (RBAC)
- *Helm Repository Management
- Multi-environment Management (multi k8s clusters, On-prem, Azure, Google, AWS, etc)
- Multi-Provider provisioning (On-prem, Azure, Google, AWS, vSphere)
- Easily scale up/down Kubernetes clusters

** New in Rancher v2.x*



Deployment Architecture – Helm

Ref: http://helm.sh

What is Helm?

Open Source Package Manager for Kubernetes through the use of Charts.

Charts help you define, install and upgrade releases for Kubernetes deployment via templates and configuration.

Why Helm?

Manage Complexity

Charts describe even the most complex apps; provide repeatable application installation, and serve as a single point of authority.

Easy Updates

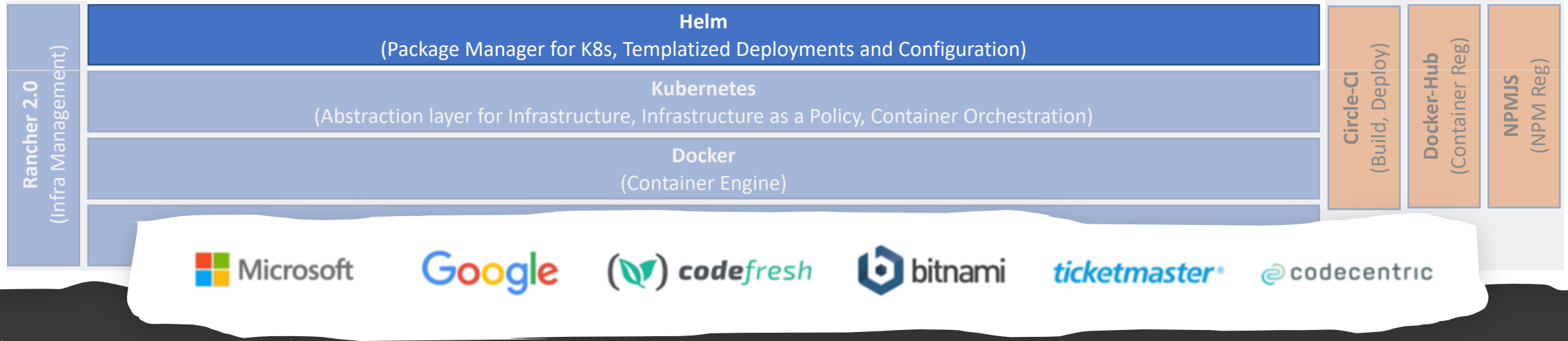
Take the pain out of updates with in-place upgrades and custom hooks.

Simple Sharing

Charts are easy to version, share, and host on public or private servers.

Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease.



Deployment Architecture – CircleCI Overview

circleci

What is CircleCI? Cloud based Continuous Integration & Deployment Platform

Ref: http://circleci.com

VCS Integration
CircleCI integrates with GitHub, GitHub Enterprise, and Bitbucket. Every time you commit code, CircleCI creates a build.

Automated Testing
CircleCI automatically tests your build in a clean container or virtual machine.

Automated Deployment
Passing builds are deployed to various environments so your product goes to market faster.

Notifications
Your team is notified if a build fails so issues can be fixed quickly.

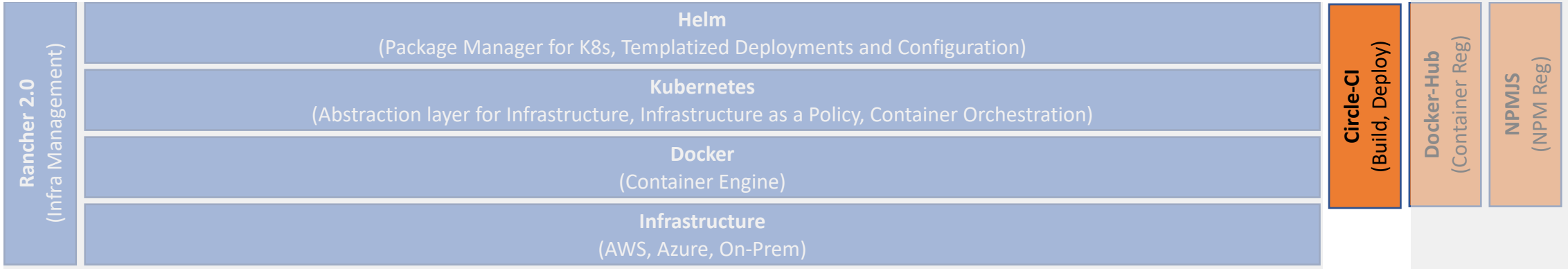
Why CircleCI?

Workflows for Job Orchestration
Orchestrate customizable job execution (such as build, test, deploy), giving complete control over your development process.

Language-Agnostic Support
Supports any language that builds on Linux or macOS, including C++, Javascript, .NET, PHP, Python, and Ruby.

First-Class Docker Support
Run any image from Docker's public/private registry or other common registries. Build Docker images, access Docker layer caching, Compose.

Powerful Caching
Speed up builds with expanded caching options, including images, source code, dependencies, and custom caches. Full control over cache save and restore points for optimal performance.



** Forrester names CircleCI a leader (https://www2.circleci.com/circleci-forrester-wave-leader-2017.html)*

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?

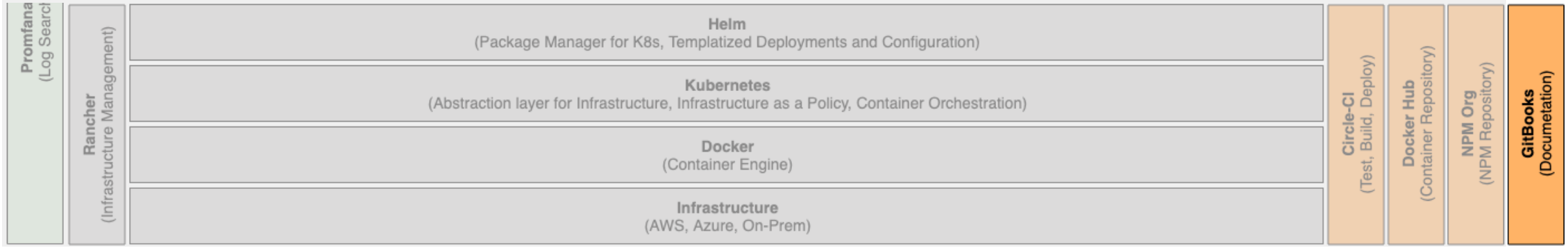
Markdown
Lightweight markup language with plain text formatting syntax supporting standard HTML, and CSS.

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.



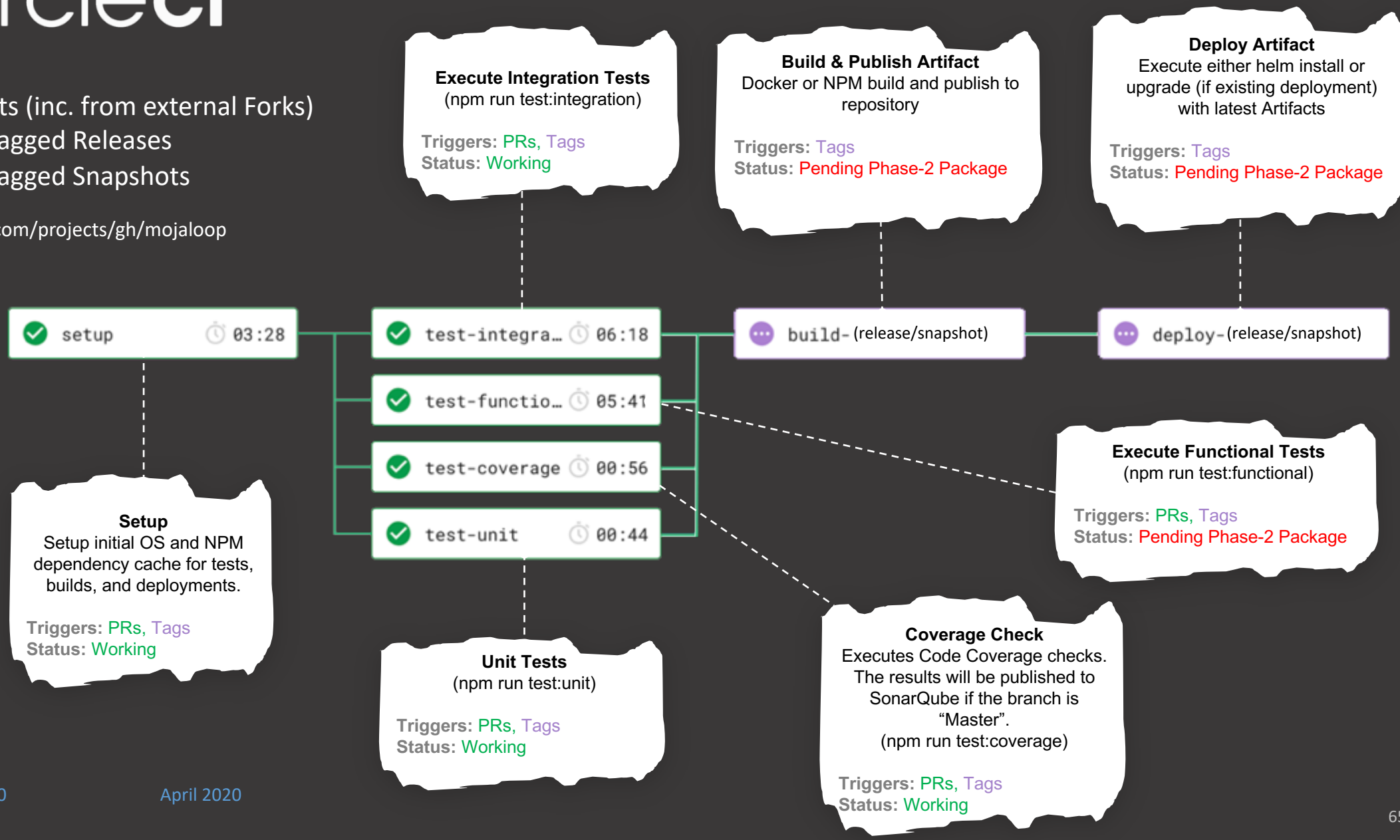
Deployment Architecture – CI/CD Pipeline

circleci

Triggers:

- [●] Pull-Requests (inc. from external Forks)
- [●] Publishing tagged Releases
- [●] Publishing tagged Snapshots

Ref: https://circleci.com/projects/gh/mojaloop



Event Monitoring Framework – SDK Example

REFERENCE

New Span:
- Generate **Trace Context** (**traceId** & **spanId**)

Finish Span:
- Close **Span**
- Record **Trace**

```
// Creates a new parent span for given service
// this sets new traceId and new spanId.
let parentSpan = Tracer.createSpan( service: 'parent service')
```

```
// Finish the span. This also sends the trace context to the tracing platform. All further operations are forbidden after the span is finished.
await parentSpan.finish(event)
```

Recorded events
- **Logs** (info, warn, debug, verbose, perf, error)
- **Audit**

```
// Logs message with logging level info from the parent span
await parentSpan.info(event)
await parentSpan.warning( message: 'event')
await parentSpan.error('event')
await parentSpan.debug('message')
await parentSpan.verbose( message: 'message')
await parentSpan.performance( message: 'message')
await parentSpan.audit( message: 'message')
// Logs message with logging level debug from the parent span
await parentSpan.debug('this is debug log')
```

Add Tags
- Add arbitrary **tags** for **metadata** as part of **Trace Context**

```
// Set tags to the span
IIChildSpan.setTags({ one: 'two' })
```

Child Span
- **Create Child Span** from a parent span
- Generate **spanId**, and set **parentId**

```
// Creates child span from the parent span with new service name.
// The traceId remains the same. The spanId is new and the parentSpanId is the spanId of the parent.
let IIChildSpan = parentSpan.getChild( service: 'child II service')
```

Trace Context
- **Inject Trace Context** into a Message
- **Extract Trace Context** from a Message
- **Create Child Span** from **Trace Context**

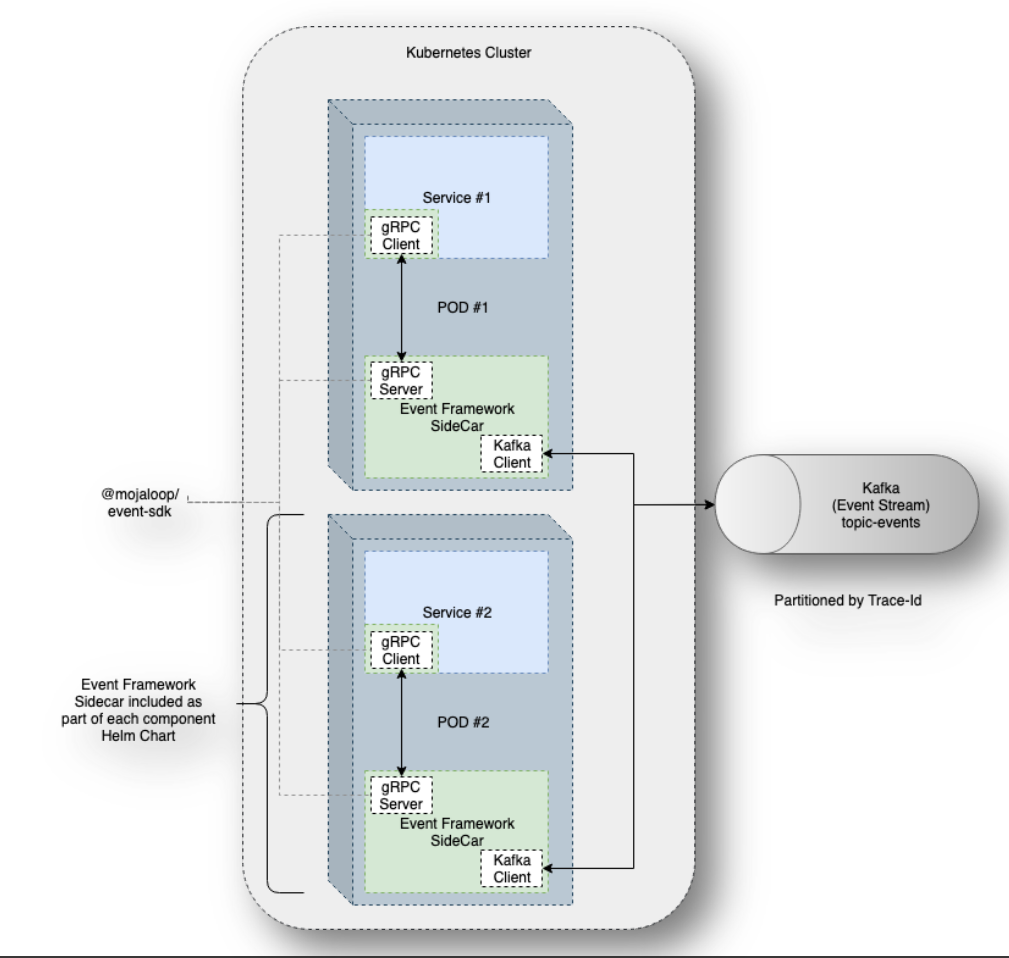
```
// Injects trace context to a message carrier. When the trace is carried across few services, the trace context can be injects in the carrier that transports the data.
let messageWithContext = await IIChildSpan.injectContextToMessage(event)
// await sleep(2000)

// Extracts trace context from message carrier. When the message is received from different service, the trace context is extracted by that method.
let contextFromMessage = Tracer.extractContextFromMessage(messageWithContext)

// Creates child span from extracted trace context.
// let IIIChild = Tracer.createChildSpanFromContext('child III service', contextFromMessage, { defaultRecorder: new DefaultLoggerRecorder() })
let IIIChild = Tracer.createChildSpanFromContext( service: 'child III service', contextFromMessage)
```

April

Event Framework – Deployment Arch & Roadmap



Deployment Architecture

Future Roadmap

1. Design considerations:
   a. Audit requirements / functionality
   b. Crypto signatures for Audit logs (single & batch)
   c. Opentracing / Zipkin dashboards if EFK is not adequate
   d. Sidecar inter-lock
2. Implementation:
   a. Audits sig & processing
   b. Official releases
   c. Mojaloop Helm Chart integration

GitHub PoC Repositories*

1. https://github.com/mojaloop/event-sdk
2. https://github.com/mojaloop/event-sidecar
3. https://github.com/mojaloop/event-stream-processor
4. https://github.com/mojaloop/apm-agent-nodejs
5. https://github.com/mojaloop/apm-agent-nodejs-opentracing
6. https://github.com/mojaloop/opentracing-javascript

** Note:*

1. *Event PoC code currently in feature branches*
2. *Snapshot releases currently available*

Event Monitoring Framework

1. Overview
2. Functionality
3. SDK Examples
4. Roadmap
5. Demo

Event Monitoring Framework – Overview

What

1. Unified framework to capture all Mojaloop events and ingest them appropriately
2. Event Types:
    a. Logs
    b. Errors
    c. Audits
    d. Traces

Why

1. Operational monitoring of requests end-to-end
    a. End-to-end request visualization
    b. Enabler for alerts
    c. Issue resolution
2. Enabler for auditing and fraud management

How

1. Standardized framework for capturing events (types, actions, metadata, etc)
2. Every request is given a trace-id at the boundary.
3. Standard common (Event-SDK) library that will publish events to a sidecar component utilising a light-weight highly performant protocol (*e.g. gRPC*)
4. Sidecar module will publish to a Kafka messaging stream for all events utilising Event-SDK
5. Each Mojaloop component will have its own tightly coupled Sidecar
6. Leverage on open source standards and solutions where possible (*e.g. APM, OpenTracing, Zipkin, etc*)

Event Framework – Functionality

REFERENCE

Event SDK Features

1. Spans
   1. Create new span
   2. Create child spans
2. Event Types:
   1. Logs (info, debug, error, performance, warning, verbose)
   2. Audit (default, start, finish, ingress, egress)
   3. Traces (Spans)
3. Client/Server:
   1. gRPC client
   2. gRPC server
4. Recorders:
   1. Async gRPC (event-sidecar)
   2. Sync gRPC (event-sidecar)
   3. Logger (Own Logger Instance)
5. Supports arbitrary tags
   1. Add additional information as part of the trace context (e.g. transferId)
6. Context
   1. Inject Context into Message
   2. Extract Context from Message
   3. Create Child Spans from Context
   4. Support for Messaging (e.g. Kafka) & HTTP Transports (WC3 Tracing *standard)
   5. Support for ”traceState” header from WC3 Tracing *Standard

* https://www.w3.org/TR/trace-context-1/

Event Sidecar

1. Publishes Messages to Kafka Event Topic

Event Stream Processor

1. Record **logs** & **audits** to **EFK**
2. Record **traces** to APM server

Helm

1. Helm Charts with Event-sidecar support:
   a. ML-API-Adapter
   b. Central-Ledger
   c. Quoting-Service
   d. SDK-Schema-Adapter
   e. Legacy Simulator

Improvements in PI8

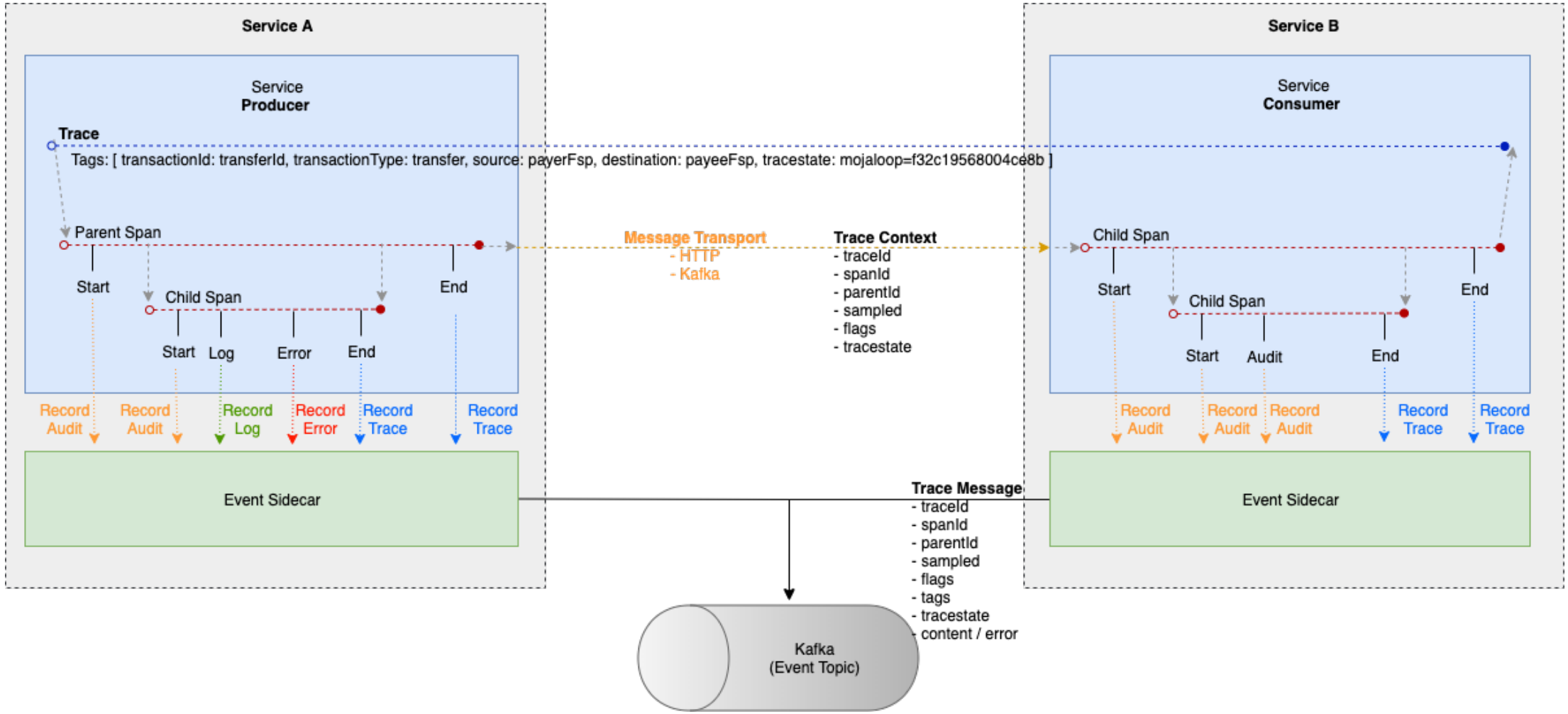
1. Enhanced WC3 Tracing Standard Support
   a. Tracestate header
2. General
   a. Sync - Async configuration for Event levels (trace, audit, log)
   b. Precise control over logging (enable/disable, local/remote & metadata)
   c. Improved stability

Future Roadmap

1. Roll-out
   a. Account-Lookup-Service
2. Performance testing
   a. What is the impact of enabling Tracing
   b. What is the optimum/acceptable level of tracing/logging

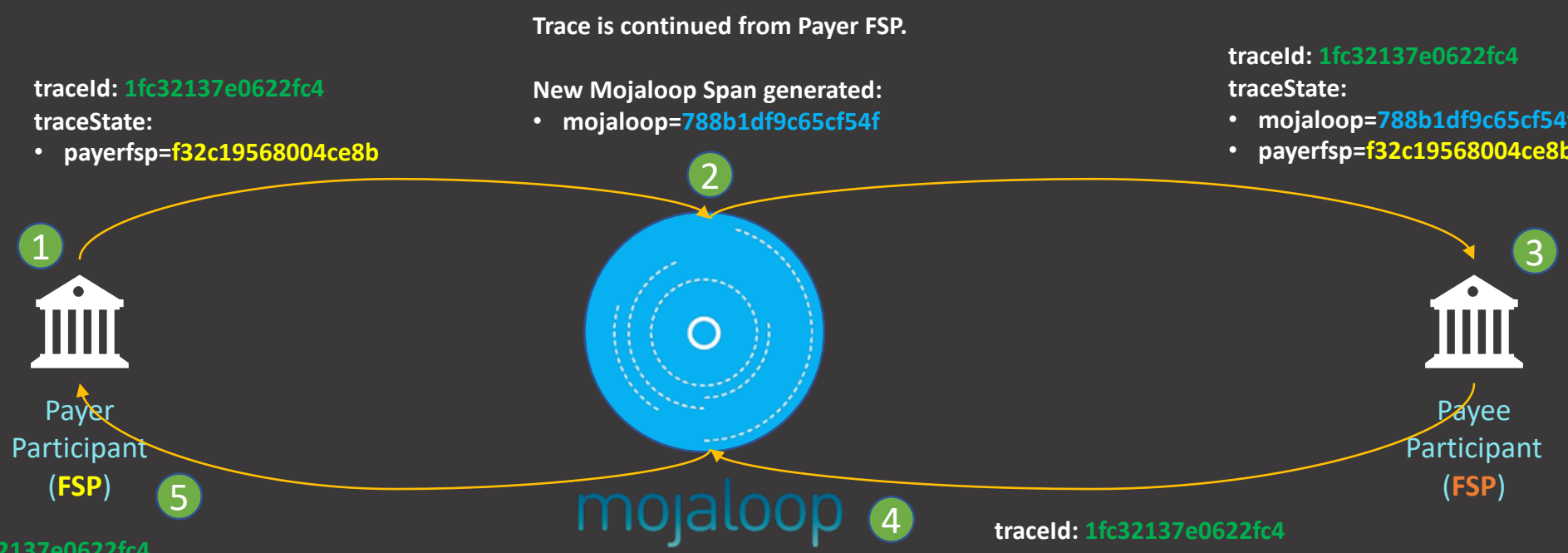
Event Monitoring Framework – Trace Architecture

REFERENCE



Event Flow of Events

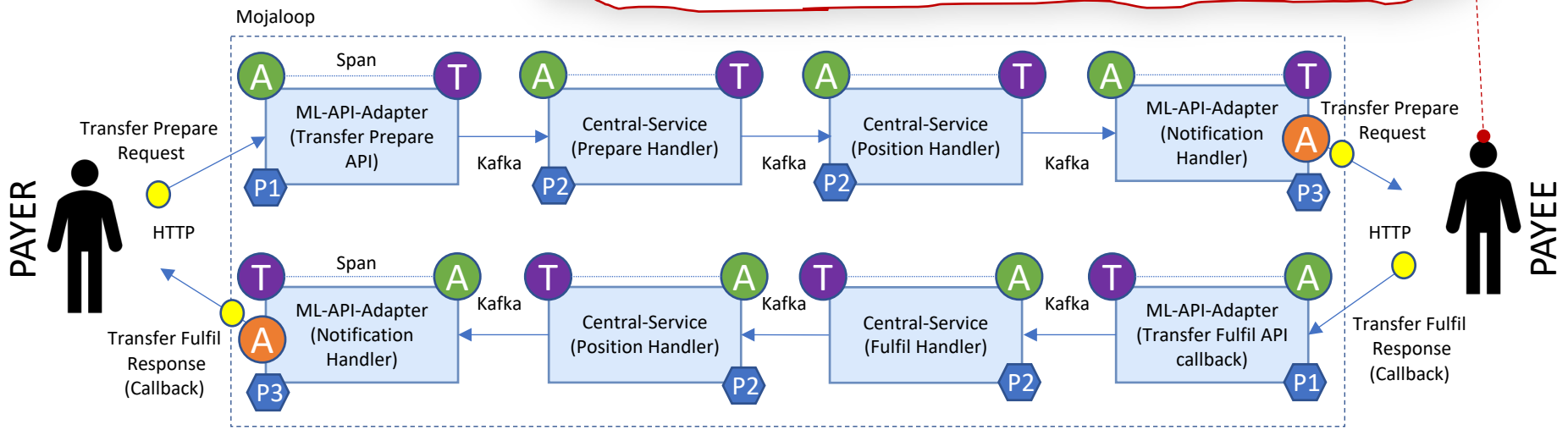
Tracing Enhancement – Federated Tracing



REF: https://www.w3.org/TR/trace-context-1/#relationship-between-the-headers

Event Monitoring Framework Use-Case Example – Success End-to-end Scenario

- Optional support for FSPs to propagate/partake in Tracing via **Wc3 Trace Headers**
- Support for **Federated Tracing** (3rd Party Vendors) via "**tracestate**" header
- Required for "end-to-end" trace **or** Prepare and Fulfil legs will be separate Traces



Event Key

- **Audit** - Start
- **Audit** - Egress
- **Trace** - Finish Span
- **Log** - Error
- **Trace-State:** Vendor Trace propagated

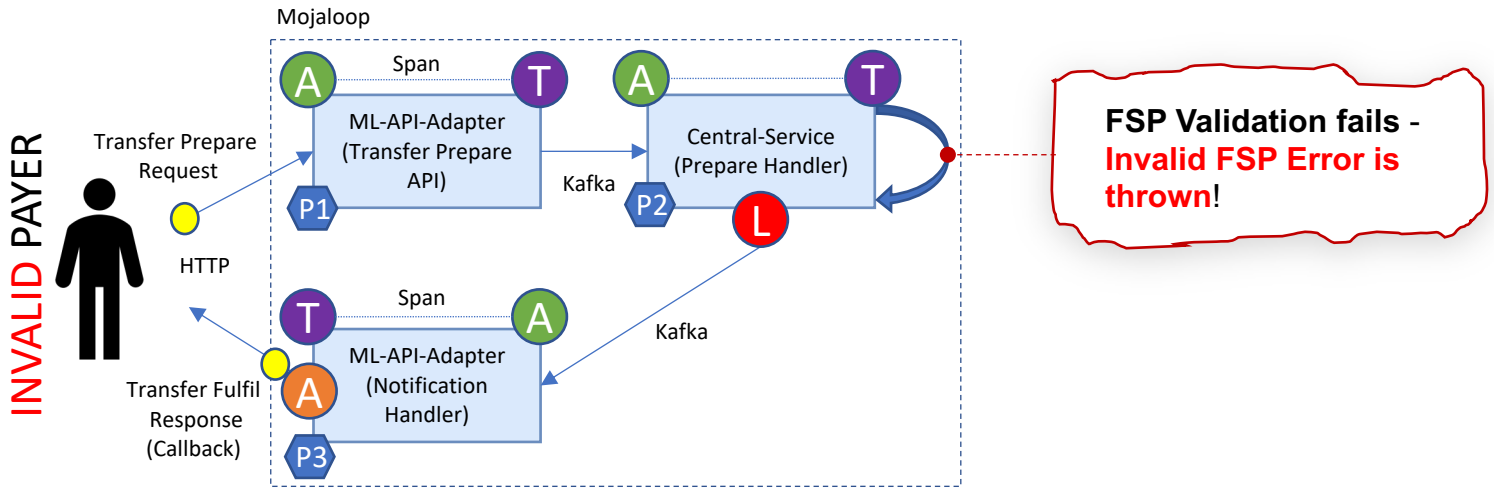
Processing Key

P1
- **Trace** is **created** with an initial Span or **continued** if **WC3 Trace Headers** are found
- **HapiJS plugin** provides consistent mechanism to manage incoming Traces
- **Trace-Context injected** into Message

P2
- **Trace-Context extracted** from message
- **Child span created**
- **Trace-Context injected** into Message

P3
- **Trace-context** is **injected** into **WC3 Trace Headers**
- **HapiJS plugin** provides consistent mechanism to manage outgoing Traces

Event Monitoring Framework Use-Case Example – Failed End-to-end Scenario



Event Key

- **Audit** - Start
- **Audit** - Egress
- **Trace** - Finish Span
- **Log** - Error
- **Trace-State:** Vendor Trace propagated

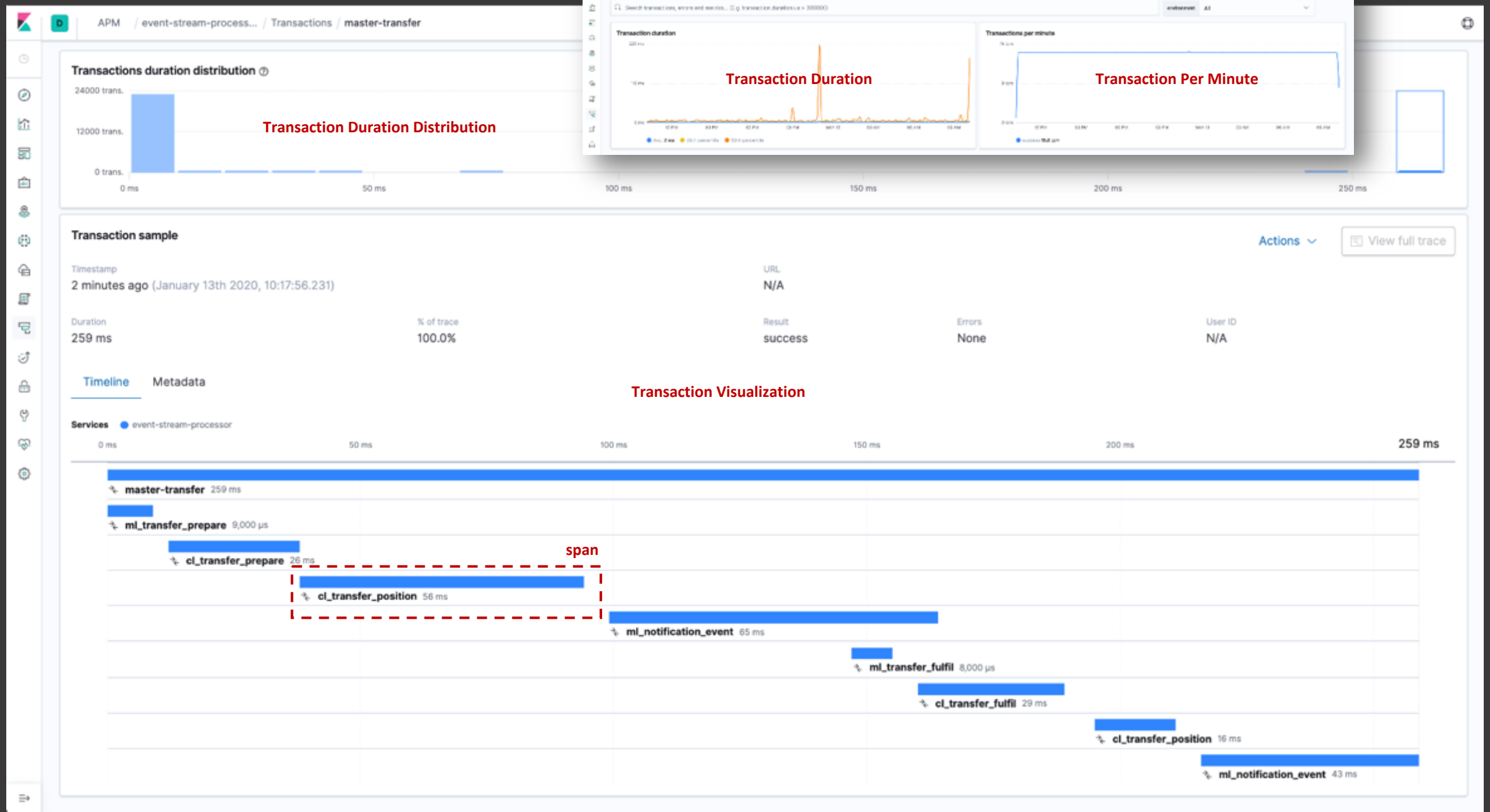
Processing Key

P1
- **Trace** is **created** with an initial Span or **continued** if **WC3 Trace Headers** are found
- **HapiJS plugin** provides consistent mechanism to manage incoming Traces
- **Trace-Context injected** into Message

P2
- **Trace-Context extracted** from message
- **Child span created**
- **Trace-Context injected** into Message

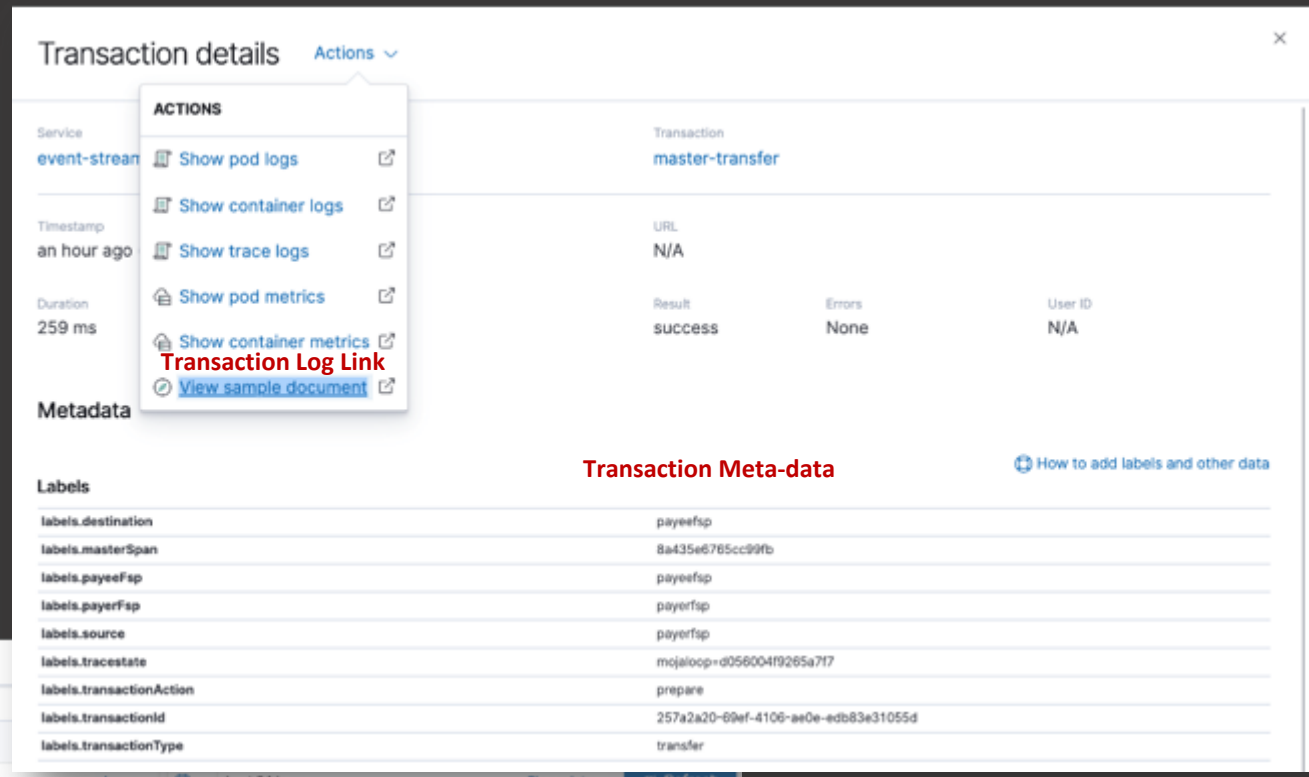
P3
- **Trace-context** is **injected** into **WC3 Trace Headers**
- **HapiJS plugin** provides consistent mechanism to manage outgoing Traces

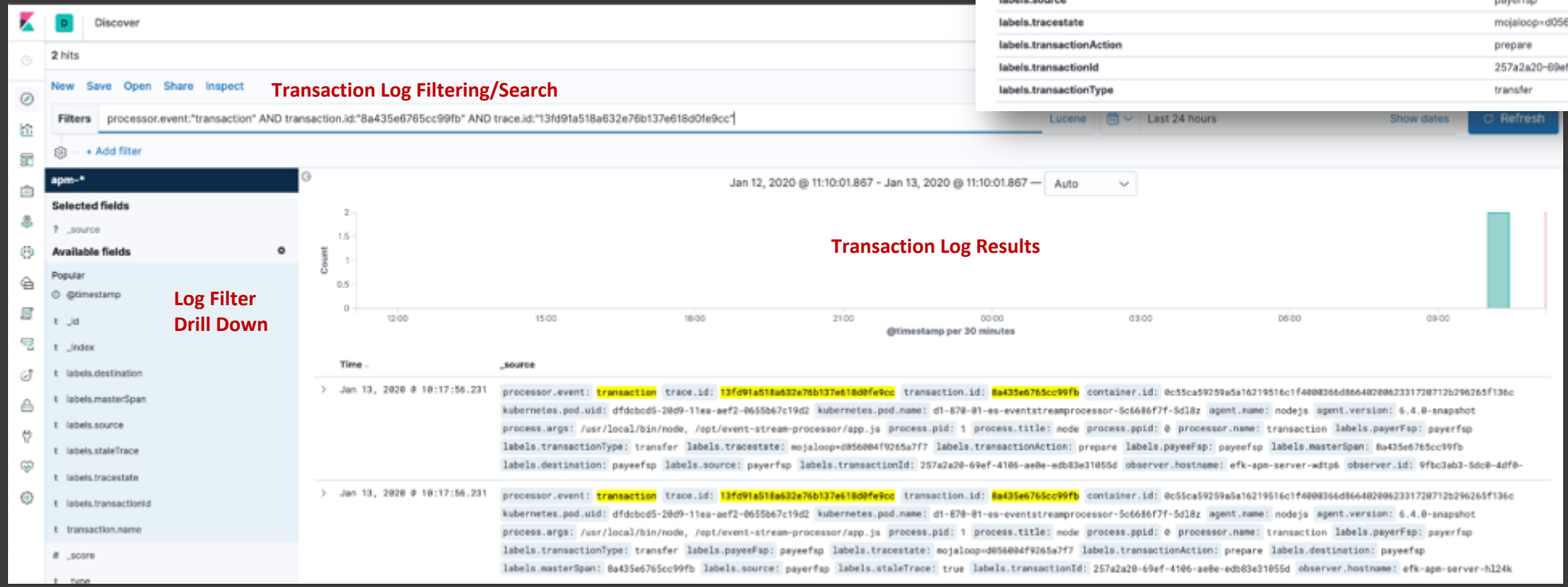
Event Monitoring Framework – Dashboard Sample



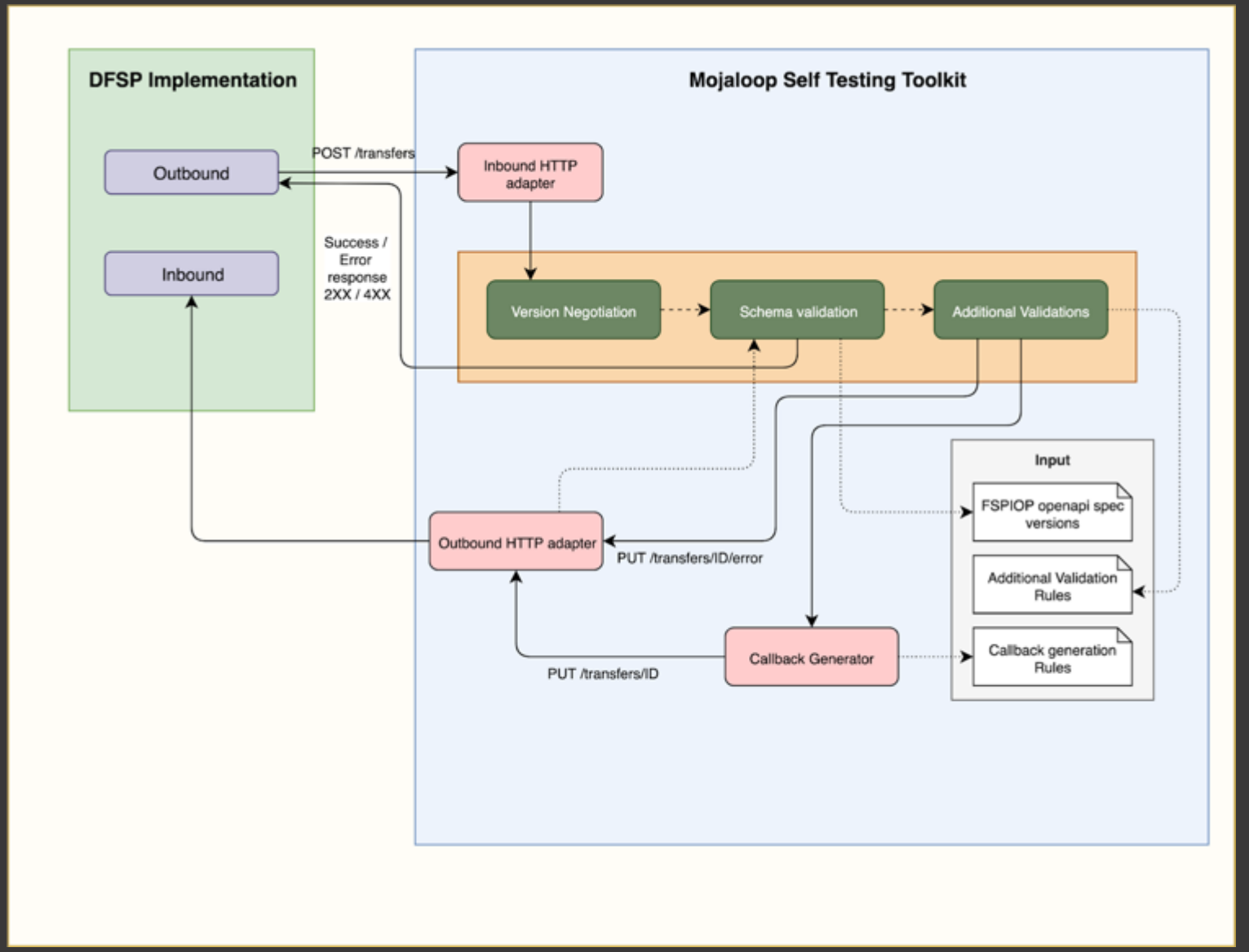
Event Monitoring Framework – Dashboard Sample (Continued)

Direct link from Tracing Dashboard to Processing & Tracing logs





Architecture Diagram – Part 1



Architecture Diagram – Part 2