Bulk Payment Enhancements

PI20 convening - progress update

Context

Why Bulk?

Core element of Pillar 2: “Help Deployments Become More Successful”

From an operator’s perspective:

- Enables a set of use cases that put money into people's wallets; revenue earners for service operators:
  - Salaries
  - Social payments
  - Loan disbursements
- Puts liquidity into people’s wallets; facilitates other use cases
- Enable governments to send social grant payments, and NGOs like World Food Programme distribute aid and support.

The DPG Ecosystem: Beyond the Needs of Operators

In support of Pillar 3

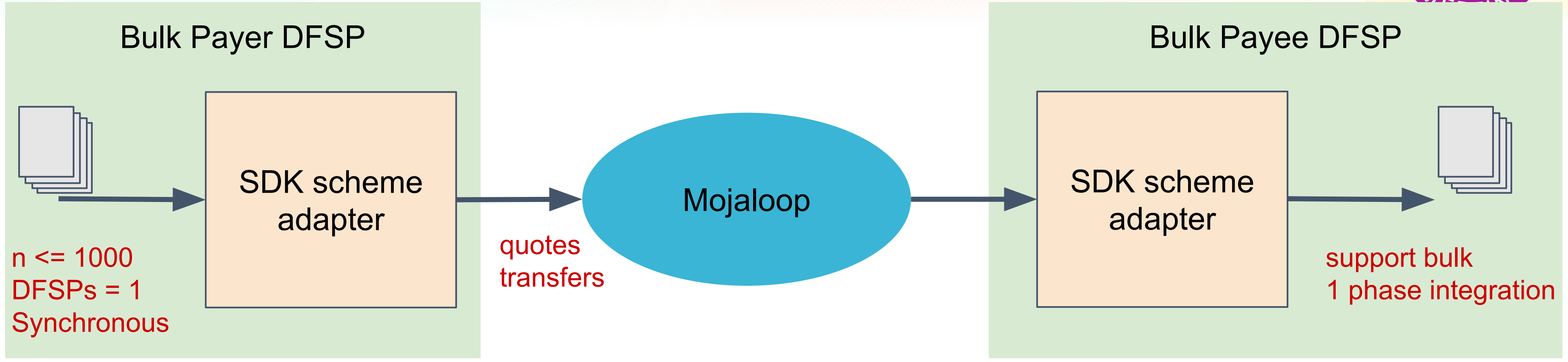
Preparing Mojaloop for integration with the wider DPG ecosystem

G2P Connect

- Integration with OpenG2P /Mifos builds on these bulk enhancements

- Integration with MOSIP a strategic imperative

What are the Enhancements enabling?



Payer DFSP Bulk Enhancements

- ✓ Larger than 1000 transaction limit
- ✓ Discovery supported in bulk transactions
- ✓ Support for multiple Payee FSPs
- ✓ Asynchronous bulk integration calling supported through SDK

Payee DFSP limitations not yet overcome

- Payee DFSP single phase integration supported

Announcement

The team has addressed some of these limitation from a Payer DFSPs perspective as part of the **Mojaloop 14.1** release (not limited to).

… and we have a demo to show you.

Bulk Enhancements

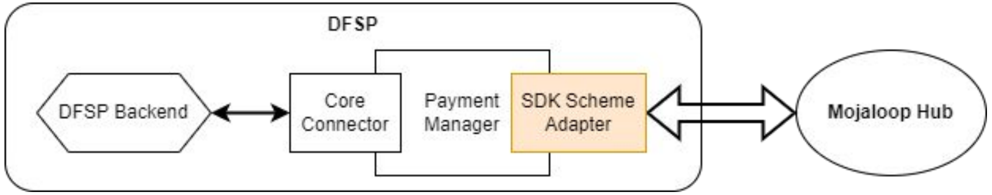
- ✓ Larger than 1000 transaction limit
- ✓ Support for multiple Payee FSPs
- ✓ Asynchronous bulk integration calling supported through SDK
- ✓ Discovery supported in bulk transactions

Distributed processing - do participants bulk processing if needed

How can these enhancements be used?

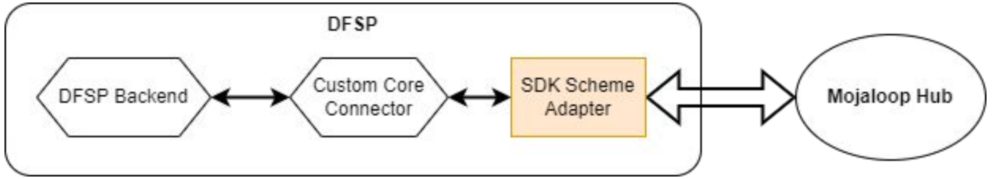
Bulk Enhancement have been built into the SDK-Scheme-Adapter as a reference implementation of best practice connection to Mojaloop

1.



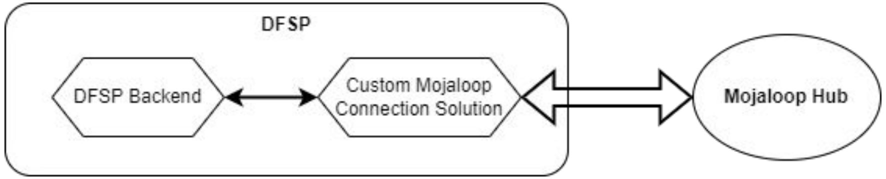
Deploy as part of third party connection product.
E.g Payment Manager in OSS

2.



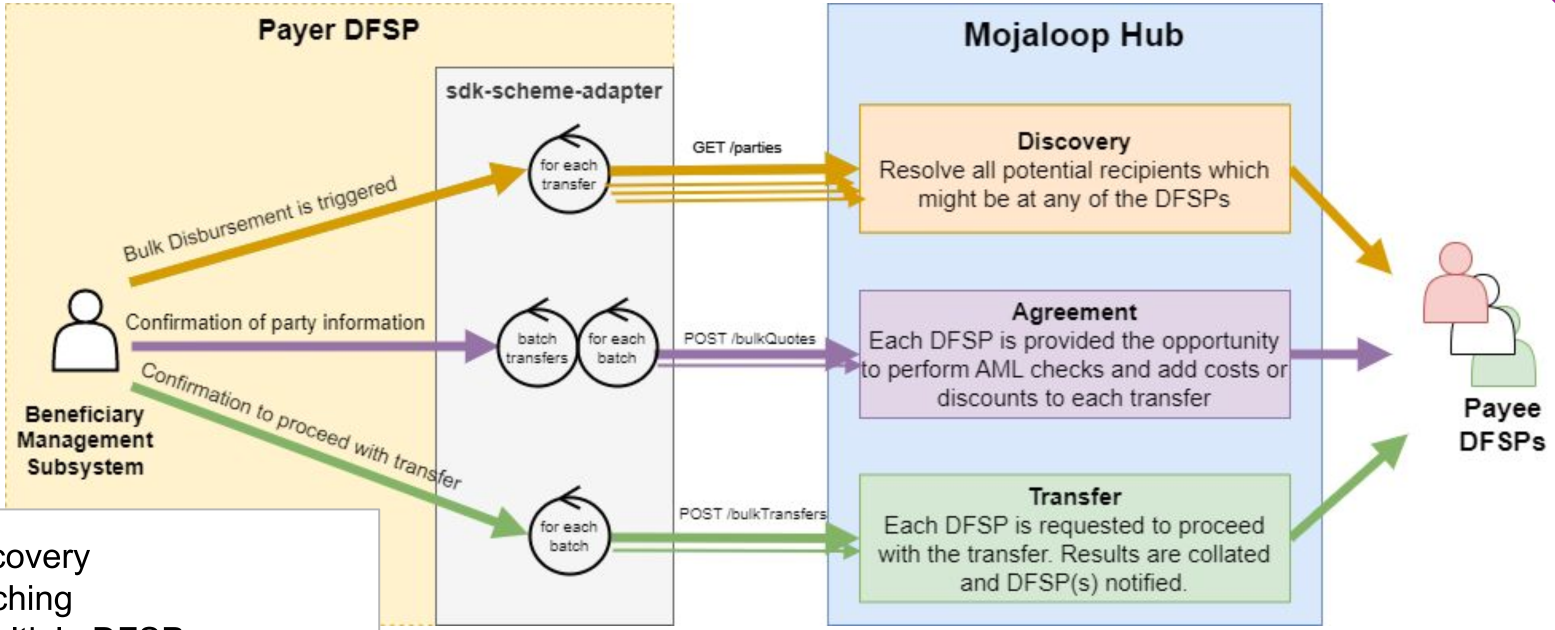
Deploy as part of custom connector.

3.



Not used directly. Can be used as a reference.
Adopt same/similar
- Interfaces - e.g. APIs & messages
- Event driven patterns
- Reuse test cases
- Use-cases - I.e. reuse the business cases

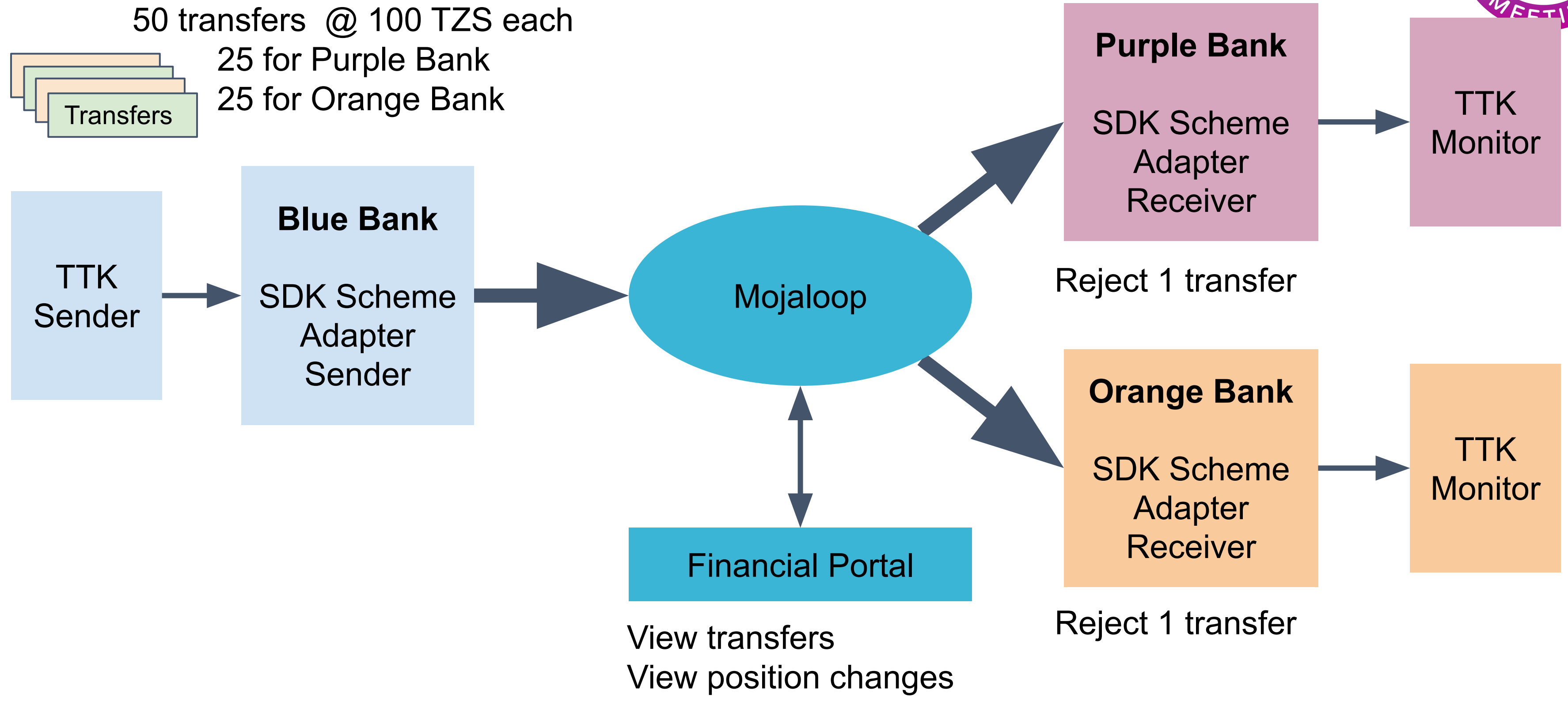
SDK - Functional design


Payer DFSP
sdk-scheme-adapter
Mojaloop Hub
Bulk Disbursement is triggered
for each transfer
GET /parties
Discovery
Resolve all potential recipients which might be at any of the DFSPs
Confirmation of party information
batch transfers
for each batch
POST /bulkQuotes
Agreement
Each DFSP is provided the opportunity to perform AML checks and add costs or discounts to each transfer
Confirmation to proceed with transfer
for each batch
POST /bulkTransfers
Transfer
Each DFSP is requested to proceed with the transfer. Results are collated and DFSP(s) notified.
Beneficiary Management Subsystem
Payee DFSPs

Add Discovery
Add Batching
- Multiple DFSPs
- Unlimited transfers

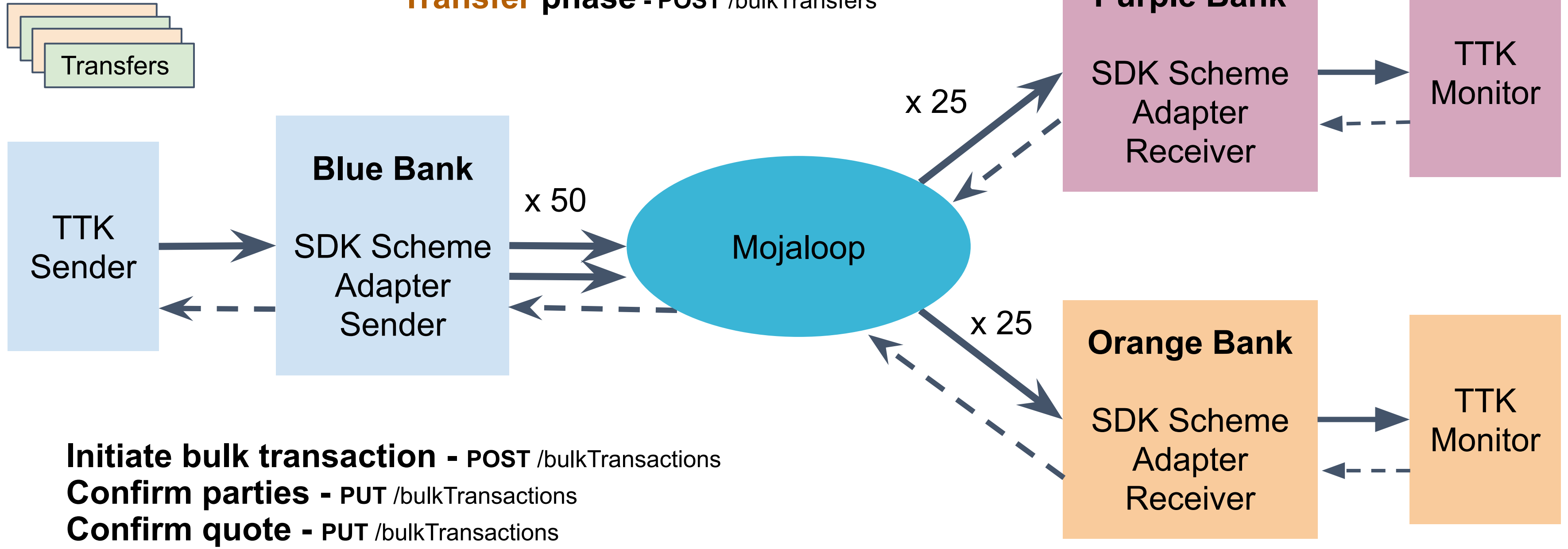
Added Asynchronous
Added flexible calling methods

Demo - Setup (Mojaloop v14.1)



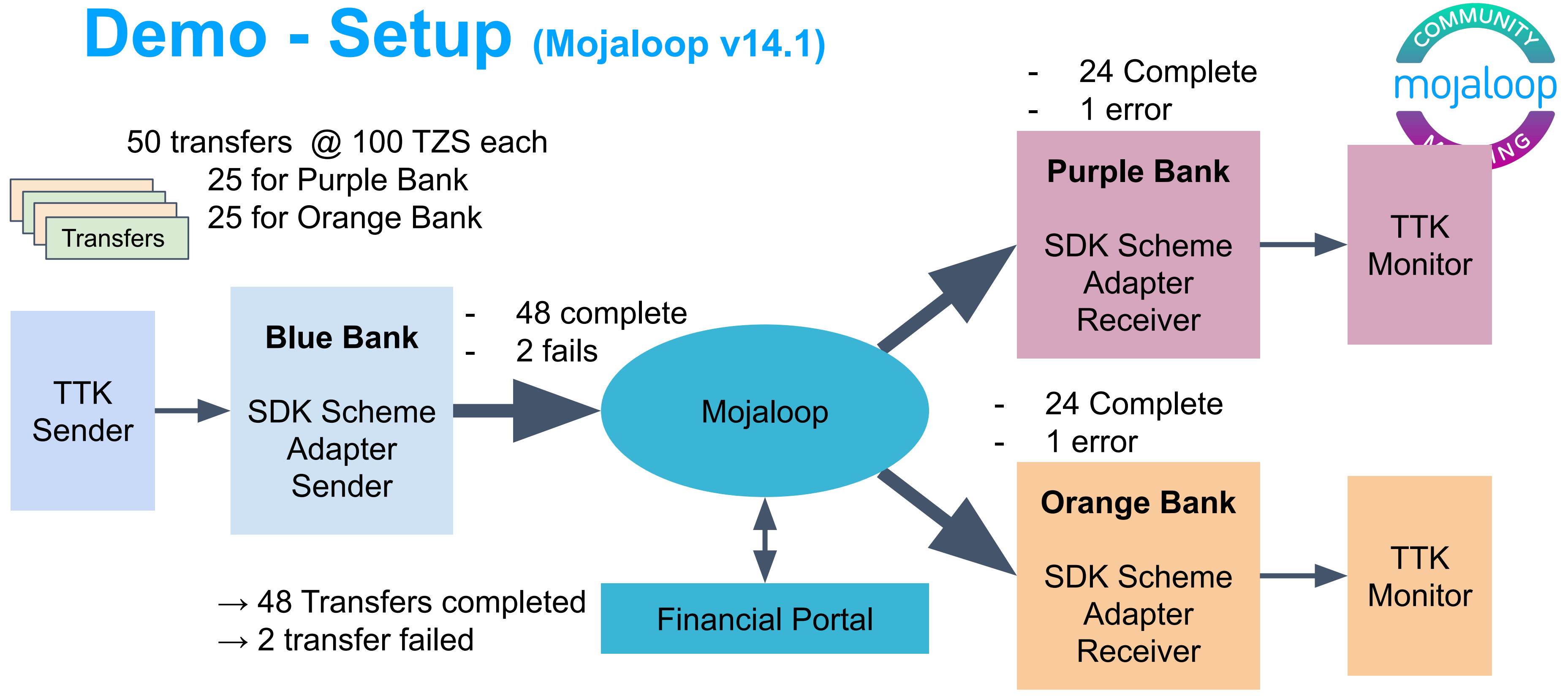
Demo - What API call are being made?

Discovery phase - GET /getParties
Agreement phase - POST /bulkQuotes
Transfer phase - POST /bulkTransfers



Initiate bulk transaction - POST /bulkTransactions
Confirm parties - PUT /bulkTransactions
Confirm quote - PUT /bulkTransactions
Receive results - PUT /bulkTransactions **callback**

Demo - Setup (Mojaloop v14.1)



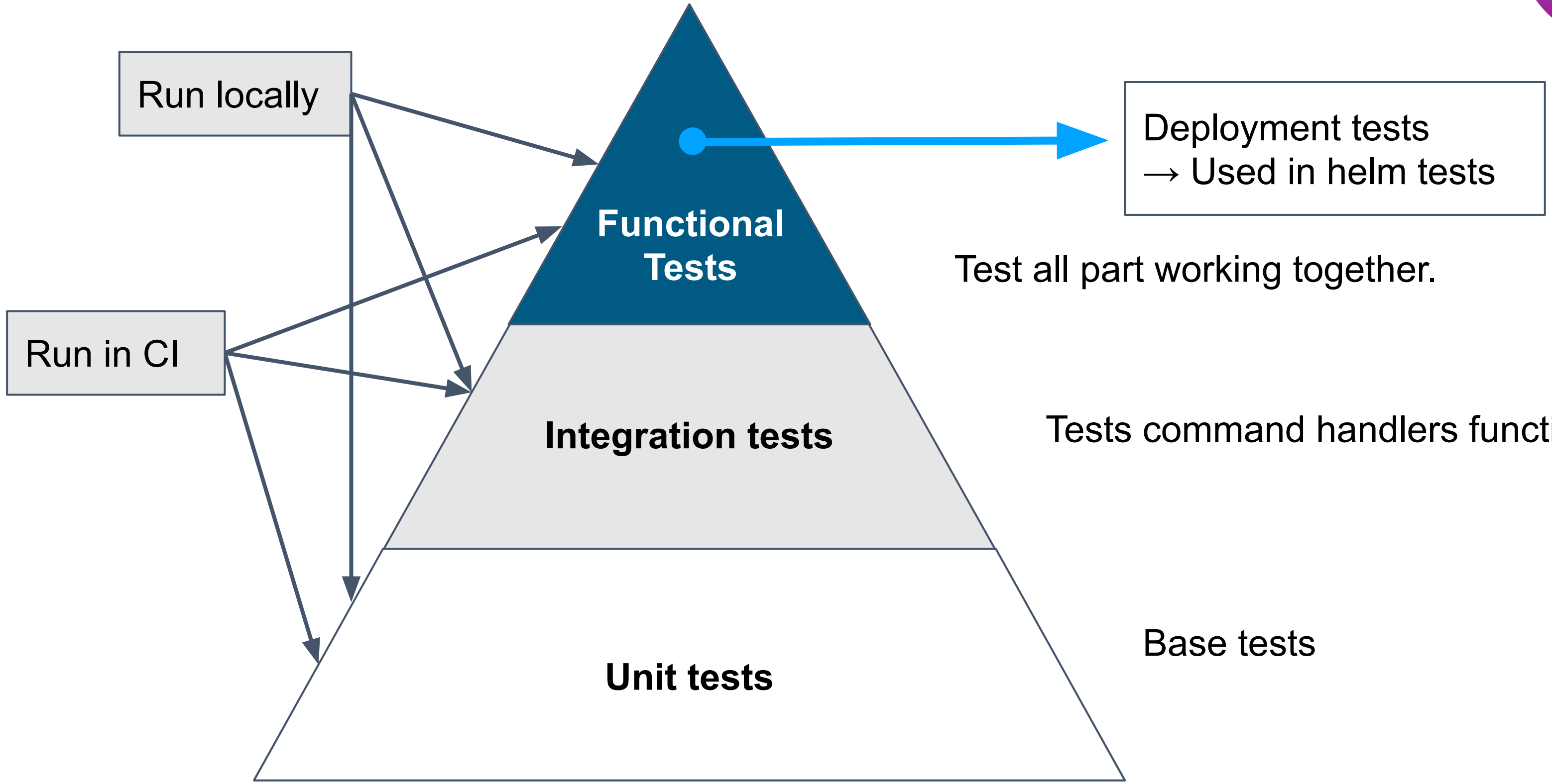
→ 48 Transfers completed
→ 2 transfer failed

Position changes

Blue Bank	- 4 800 TZS
Purple Bank	+ 2 400 TZS
Orange Bank	+ 2 400 TZS

Demo

Testing strategy taken



Links:

Integration tests:
https://github.com/mojaloop/sdk-scheme-adapter/blob/mvp/bulk-sdk/modules/outbound-command-eve
nt-handler/test/integration/application

Functional end to end tests:

https://github.com/mojaloop/sdk-scheme-adapter/tree/mvp/bulk-sdk/test/func/ttk-testcases

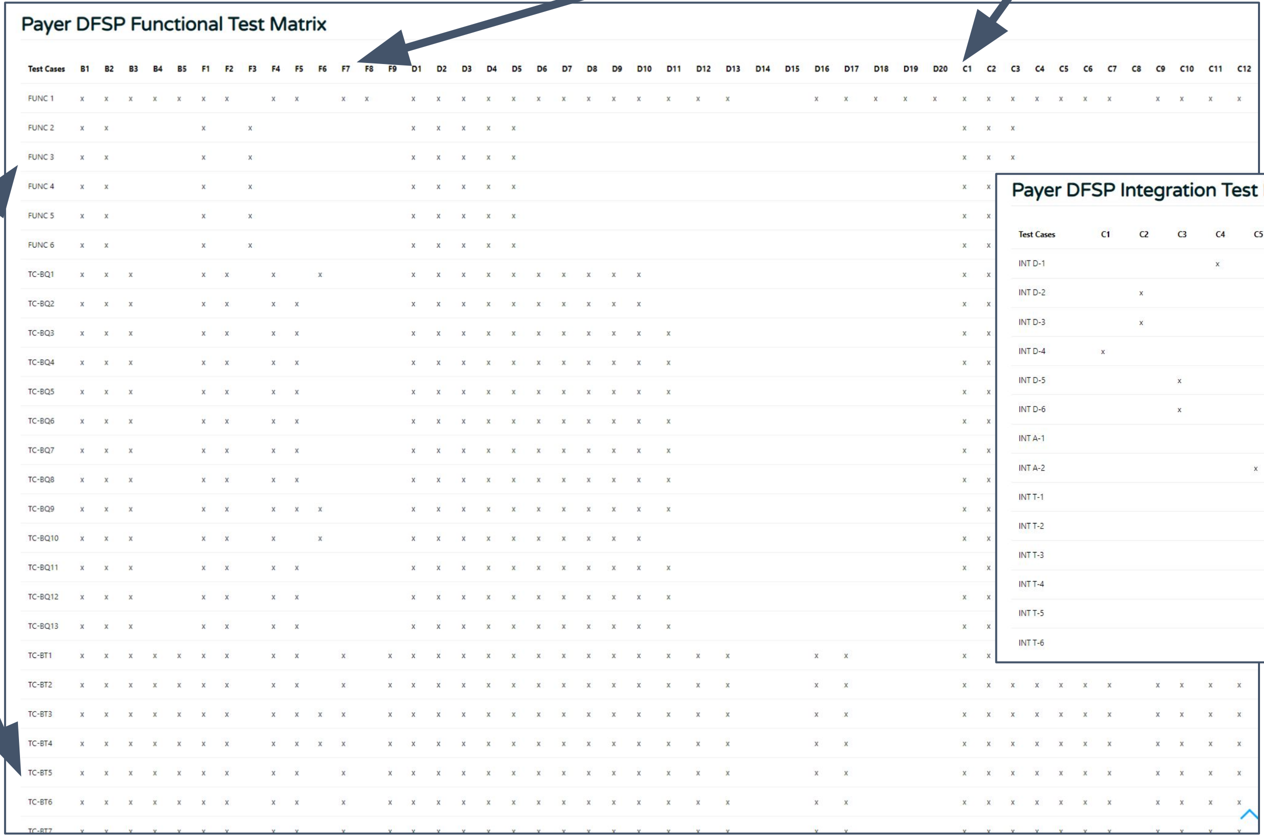
Test cases:

https://github.com/mojaloop/sdk-scheme-adapter/blob/mvp/bulk-sdk/test/func/bulk-test-cases.md

Testing Matrix

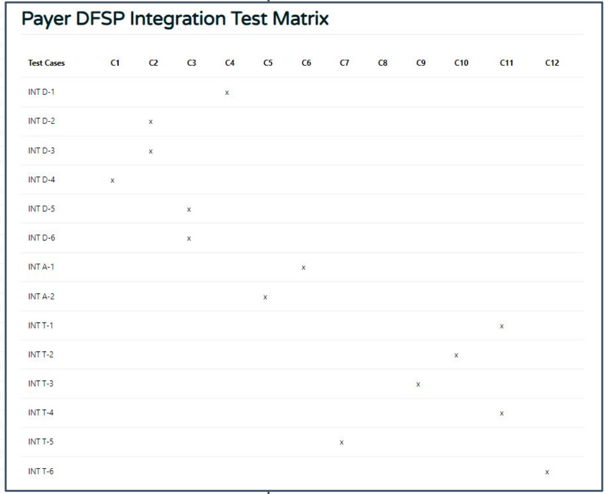
Functionality

Test cases



Payer DFSP Functional Test Matrix

Test Cases	B1	B2	B3	B4	B5	F1	F2	F3	F4	F5	F6	F7	F8	F9	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
FUNC 1	x	x	x	x	x	x	x		x	x		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
FUNC 2	x	x				x		x							x	x	x	x	x																x	x	x									
FUNC 3	x	x				x		x							x	x	x	x	x																x	x	x									
FUNC 4	x	x				x		x							x	x	x	x	x																x	x										
FUNC 5	x	x				x		x							x	x	x	x	x																x	x										
FUNC 6	x	x				x		x							x	x	x	x	x																x	x										
TC-BQ1	x	x	x			x	x		x		x				x	x	x	x	x	x	x	x	x	x											x	x										
TC-BQ2	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x											x	x										
TC-BQ3	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ4	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ5	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ6	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ7	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ8	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ9	x	x	x			x	x		x	x	x				x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ10	x	x	x			x	x		x		x				x	x	x	x	x	x	x	x	x	x											x	x										
TC-BQ11	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ12	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BQ13	x	x	x			x	x		x	x					x	x	x	x	x	x	x	x	x	x	x										x	x										
TC-BT1	x	x	x	x	x	x	x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x										
TC-BT2	x	x	x	x	x	x	x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x
TC-BT3	x	x	x	x	x	x	x		x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x
TC-BT4	x	x	x	x	x	x	x		x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x
TC-BT5	x	x	x	x	x	x	x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x
TC-BT6	x	x	x	x	x	x	x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x
TC-BT7	x	x	x	x	x	x	x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x				x	x	x	x	x	x	x		x	x	x	x



Payer DFSP Integration Test Matrix

Test Cases	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
INT D-1				x								
INT D-2		x										
INT D-3		x										
INT D-4	x											
INT D-5			x									
INT D-6			x									
INT A-1						x						
INT A-2					x							
INT T-1											x	
INT T-2										x		
INT T-3									x			
INT T-4											x	
INT T-5							x					
INT T-6												x

Architecture

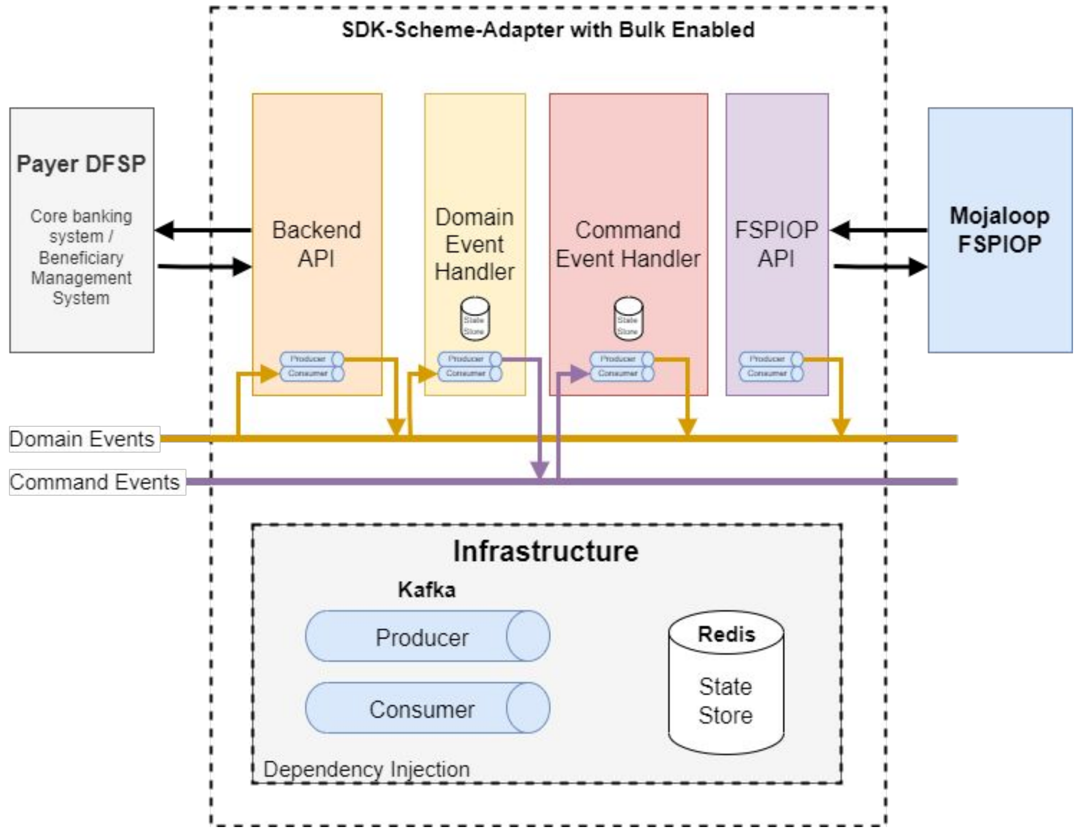
1. Designed for…
   a. Interoperability with existing functionality with minimal impact
   b. Asynchronous processing
   c. Failures - *not yet tested*
   d. Scalability - *not yet tested*

2. Customise for your use-case
   a. Lightweight fast with minimal persistence and quick deployment
   b. Production system with full redundancy
   c. Bulk-Enhancement functionality is optional

3. Tested infrastructure using Kafka and Redis

4. Kubernetes ready

Architecture Diagram



Team Contributions from

Kevin Leyow

Juan Correa

Miguel de Barros

Sam Kummary

Shashikant Hirugade

Sridevi Miriyala

Vijay Kumar

Yevhen Kyriukha

Mojaloop Bulk Feature Roadmap

	Mojaloop Hub		DFSP Connection Support			Integration Support
	FSPIOP	UI support	Payer DFSP Enhancements	Payee DFSP Enhancements	UI support	STD Core Connectors
Current features	1. bulkQuotes 2. bulkTransfers 3. Enhanced Tests - Improved test coverage		1. Asynchronous 2. >1000 transactions 3. Discovery 4. Multi - Payee DFSP 5. Persistence 6. Scalable 7. Robust (failure design)	1. SDK support for bulk Quotes 2. SDK Support for bulk Transfers		1. \bulkTransactions API Published 2. Bulk Integration patterns are documented
Road map features	1. Enable reserve - commit integrations (Bulk Patch Notification) 2. PISP Bulk	1. Bulk Transfer Dashboard (micro-frontend)	1. Synchronous 2. Auto accept party 3. Auto-accept quote (with fee limits per currency) 4. Only Validate Party 5. Skip Party lookup 6. Bulk Expiration	1. Demultiplexing for bulk Quotes 2. Demultiplexing for bulk Transfers	1. Payment Manager Support 2. Payer Bulk Transfer Dashboard	1. Hackathon 2. Bulk Mojaloop Training Program new course and course extensions
	Supporting Bulk at Hub		Supporting Bulk at DFSP			Supporting Integrations

References

SDK-Scheme-Adapter Overview

- Integration Flow Patterns
- Bulk Integration Flow Patterns

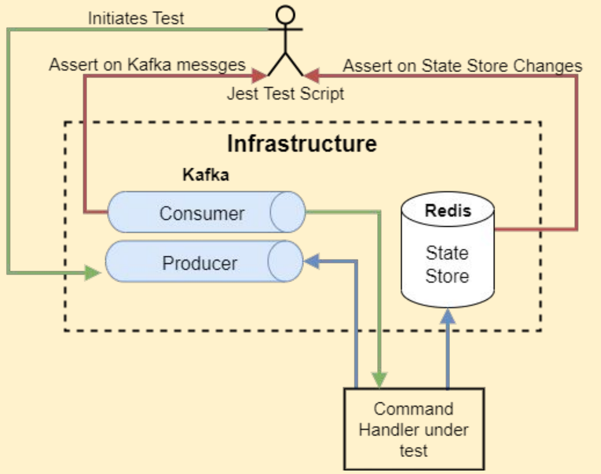
Bulk SDK-Scheme-Adapter Design Support

- API Design
  Detailed sequence diagram & error codes tables
- DDD and Event Sourcing Design
  Detailed event sourcing sequence diagrams
- Tests

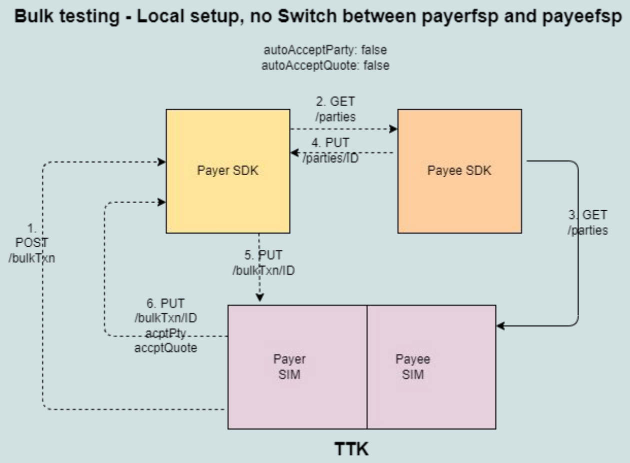
Appendix

Testing harnesses

Integration Tests
Command Handler



Functional Tests



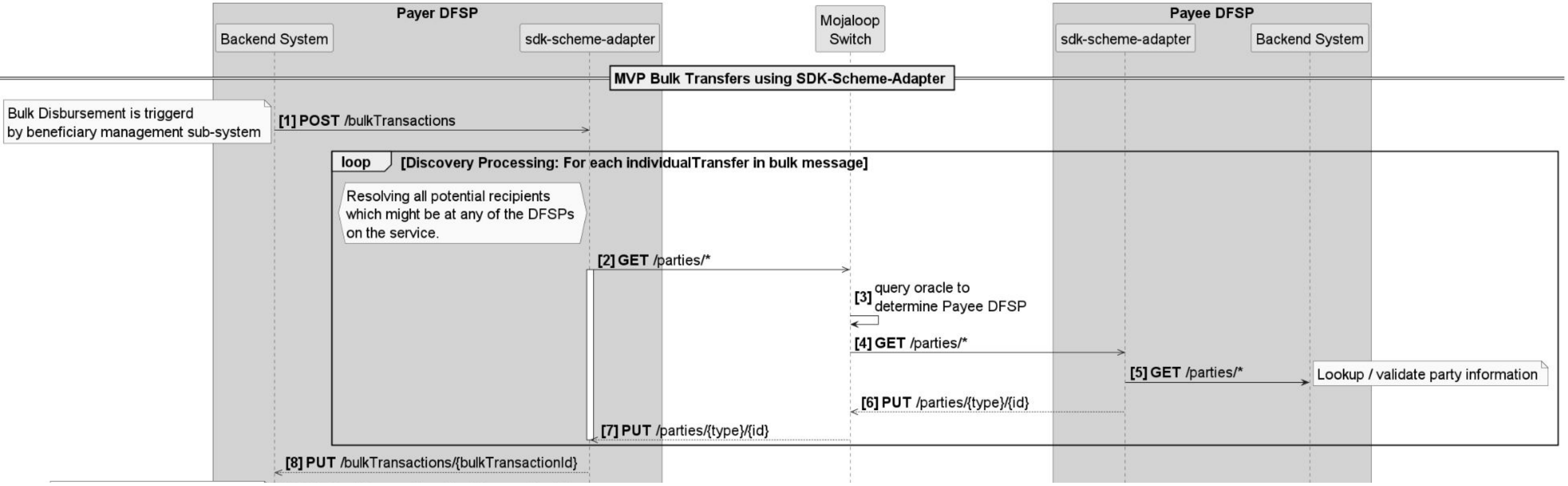
Standard functions on each module

Mono-repo testing

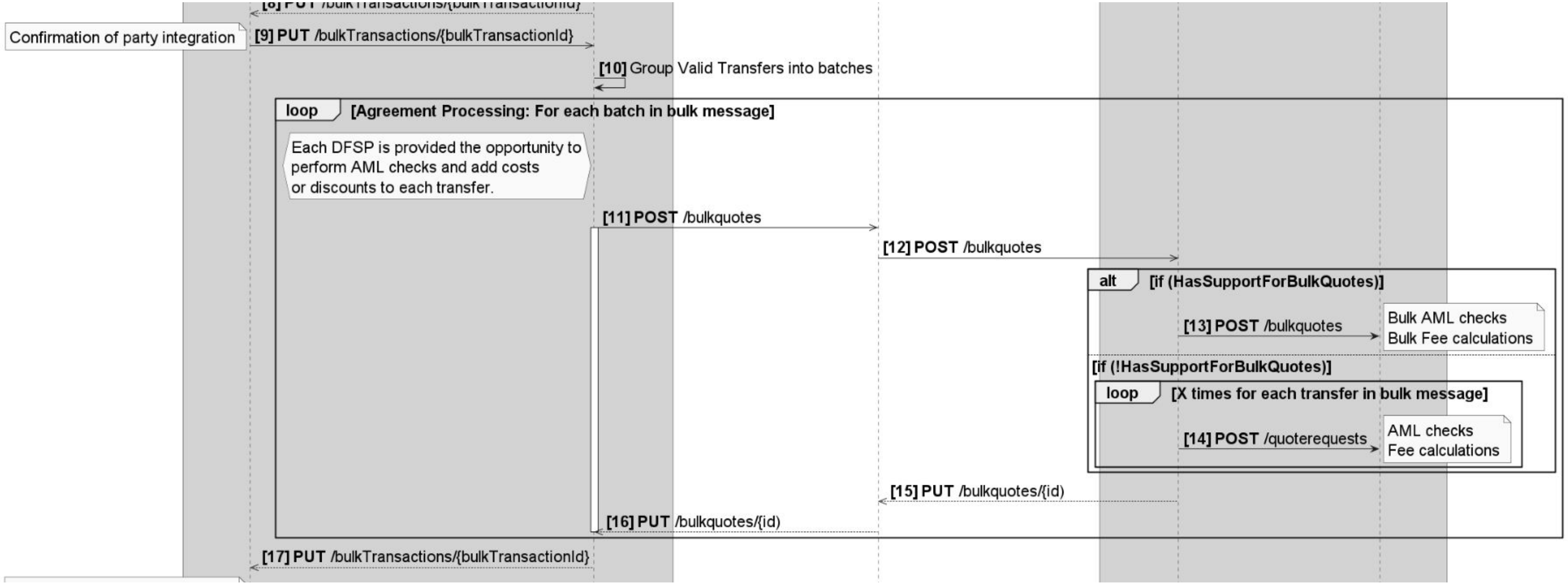
CI - Integration

Helm tests

Bulk Discovery Phase

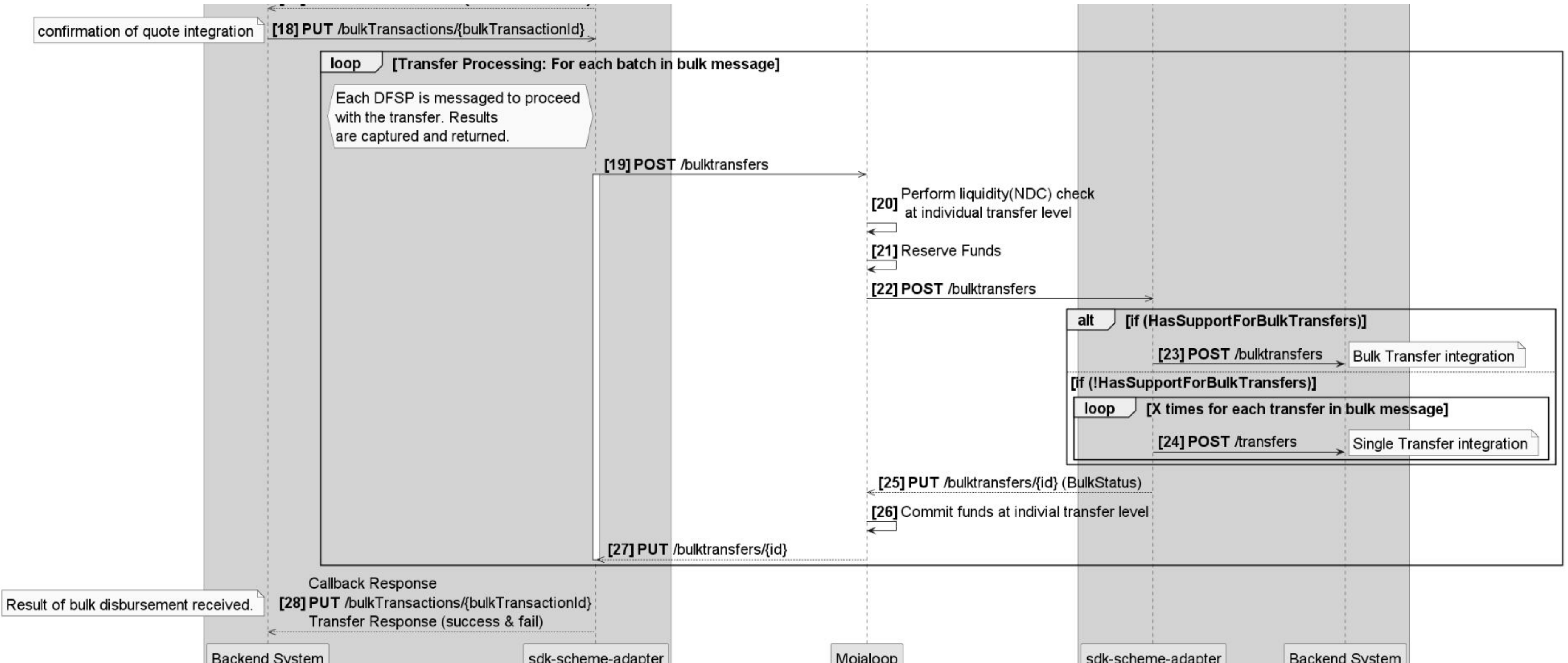

Payer DFSP
Backend System
sdk-scheme-adapter
Mojaloop Switch
Payee DFSP
sdk-scheme-adapter
Backend System
MVP Bulk Transfers using SDK-Scheme-Adapter
Bulk Disbursement is triggerd by beneficiary management sub-system
[1] POST /bulkTransactions
loop
[Discovery Processing: For each individualTransfer in bulk message]
Resolving all potential recipients which might be at any of the DFSPs on the service.
[2] GET /parties/*
[3] query oracle to determine Payee DFSP
[4] GET /parties/*
[5] GET /parties/*
Lookup / validate party information
[6] PUT /parties/{type}/{id}
[7] PUT /parties/{type}/{id}
[8] PUT /bulkTransactions/{bulkTransactionId}

Bulk Agreement Phase



[8] PUT /bulkTransactions/{bulkTransactionId}
Confirmation of party integration
[9] PUT /bulkTransactions/{bulkTransactionId}
[10] Group Valid Transfers into batches
loop [Agreement Processing: For each batch in bulk message]
Each DFSP is provided the opportunity to perform AML checks and add costs or discounts to each transfer.
[11] POST /bulkquotes
[12] POST /bulkquotes
alt [if (HasSupportForBulkQuotes)]
[13] POST /bulkquotes
Bulk AML checks
Bulk Fee calculations
[if (!HasSupportForBulkQuotes)]
loop [X times for each transfer in bulk message]
[14] POST /quoterequests
AML checks
Fee calculations
[15] PUT /bulkquotes/{id}
[16] PUT /bulkquotes/{id}
[17] PUT /bulkTransactions/{bulkTransactionId}

Bulk Transfers Phase



confirmation of quote integration
[18] PUT /bulkTransactions/{bulkTransactionId}
loop [Transfer Processing: For each batch in bulk message]
Each DFSP is messaged to proceed with the transfer. Results are captured and returned.
[19] POST /bulktransfers
[20] Perform liquidity(NDC) check at individual transfer level
[21] Reserve Funds
[22] POST /bulktransfers
alt [if (HasSupportForBulkTransfers)]
[23] POST /bulktransfers
Bulk Transfer integration
[if (!HasSupportForBulkTransfers)]
loop [X times for each transfer in bulk message]
[24] POST /transfers
Single Transfer integration
[25] PUT /bulktransfers/{id} (BulkStatus)
[26] Commit funds at indivial transfer level
[27] PUT /bulktransfers/{id}
Callback Response
[28] PUT /bulkTransactions/{bulkTransactionId}
Transfer Response (success & fail)
Result of bulk disbursement received.
Backend System
sdk-scheme-adapter
Mojaloop
sdk-scheme-adapter
Backend System