mojaloop

Cross Network Mojaloop

Sending payments between Mojaloop systems

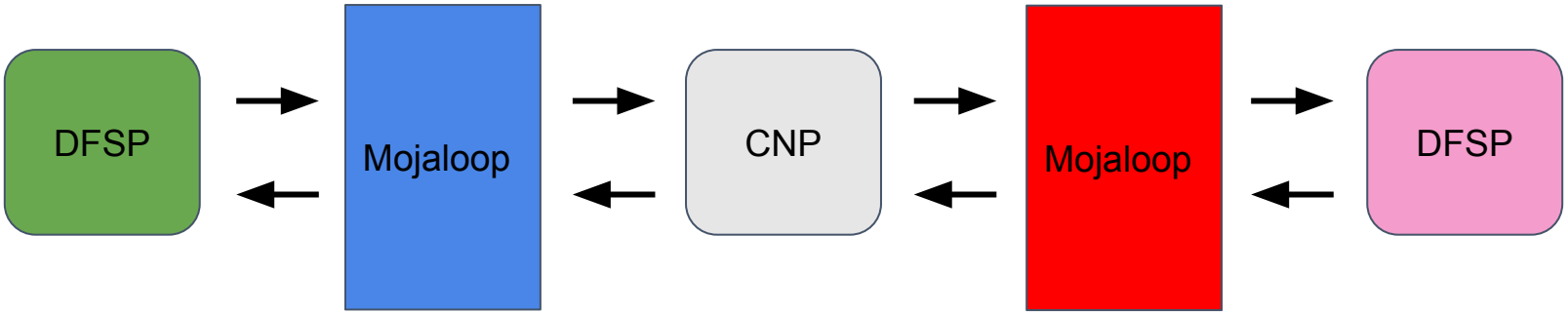
Proof of Concept - Part 1

Agenda

- Goals and Part 1 Scope
- Ecosystem
- Design Decisions
- Current Design
- Caveats
- How It Works
- Note on In-Country FX
- Community Contributions
- Proposed API changes
- Next Steps

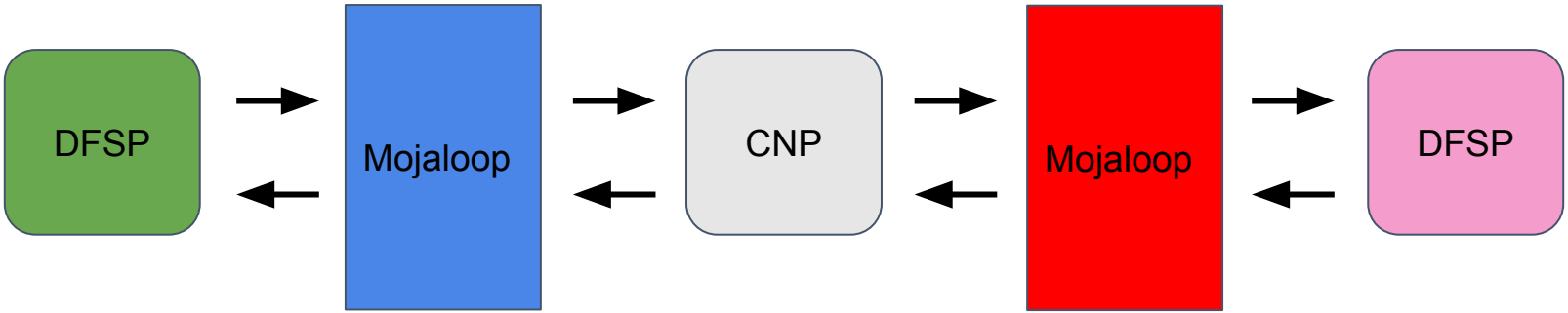
Goals of the POC

Demonstrate that a payment can be sent from a DFSP on one Mojaloop network to a DFSP on another Mojaloop network using a special DFSP (a **cross-network provider**) as a gateway.

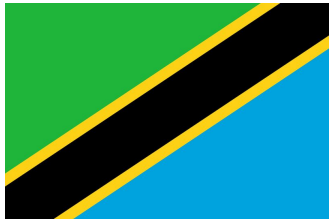


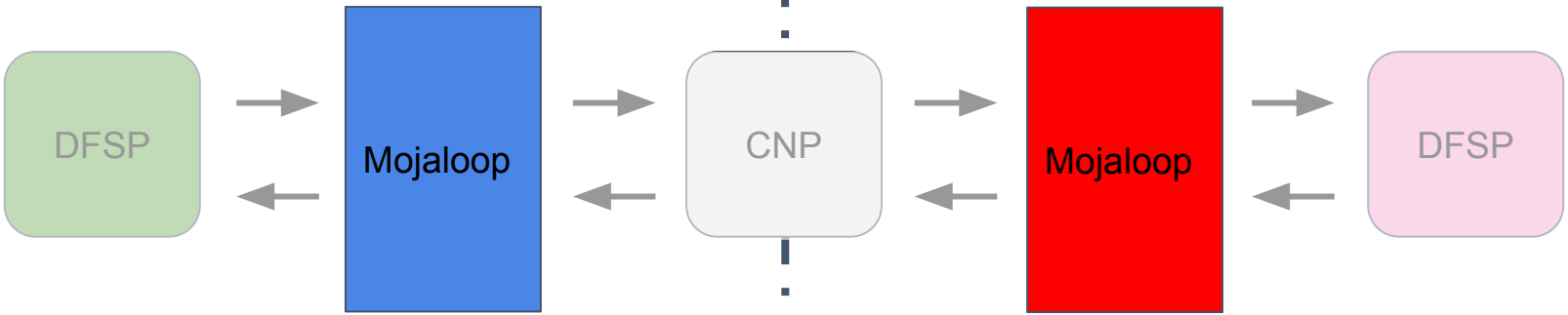
Part 1 Scope

1. Lookup payee (Moja Address)
2. Send a cross-network quote
3. Send a cross-network transfer (single currency)



Ecosystem

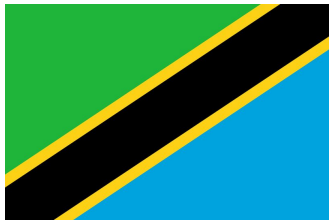


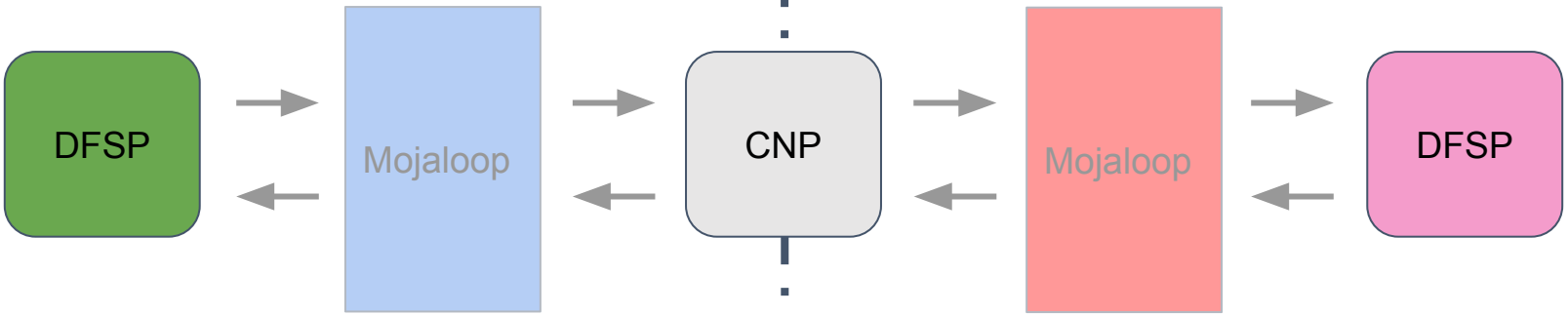


Blue Moja
Domestic Mobile Money in South Africa

Red Moja
Domestic Mobile Money in Tanzania

Ecosystem





Green Mobile
DFSP in South Africa

Super Remit
Remittance provider operating in South Africa and Tanzania

Pink Mobile
DFSP in Tanzania

Design Decisions

- No changes at the DFSP
- Use a **cross-network addressing scheme**
- **Routing logic** implemented at the centre

Current Design

- Use an Interledger Protocol-based addressing scheme for DFSPs
- Custom **moja** allocation scheme (Moja Addresses):
  e.g. *moja.tz.red.tsh.pink*
- Return *Moja Address* as *FspId* during lookup
- All API calls go via Mojaloop to be routed appropriately
- Interop Switch and Moja API Adaptor consult routing service to set correct destination headers
- Mock CNP is an Interledger Connector with Moja API plugins
  (API calls mapped to ILP packets internally)

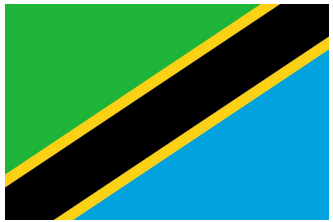
Caveats

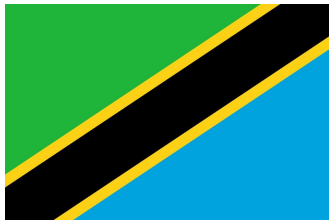
- URL-based lookup system won't work outside VPNs unless endpoints are on public Internet
- Endpoint data for *interop-switch-js* (lookup and quote APIs) is stored in Central Ledger's DB to re-use existing schema. Should be stored locally (in Interop Switch service DB) or accessed via admin API on Central Ledger.
- Routing logic is behind an API endpoint on the Central Ledger. Should be implemented in a stand-alone routing service which exchanges routing data with peers.

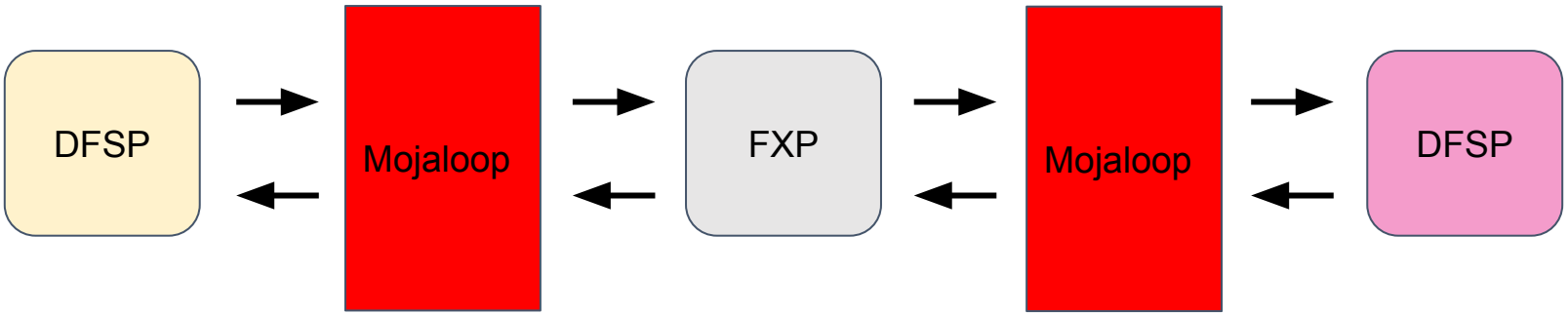
How It Works

1. Sending DFSP performs a **lookup** and gets a Moja Address as the value of the *FspId*
2. DFSP sends a **quote** to interop-switch-js which resolves a route for the API call and forwards the quote to the CNP
3. CNP forwards quote to the payee DFSP via interop-switch-js in payee's network and routes response back
4. DFSP sends a **transfer** to ml-api-adapter which resolves the correct local DFSP via a routing API call and puts transfer on the correct queue on the ledger.

Note on In-Country FX





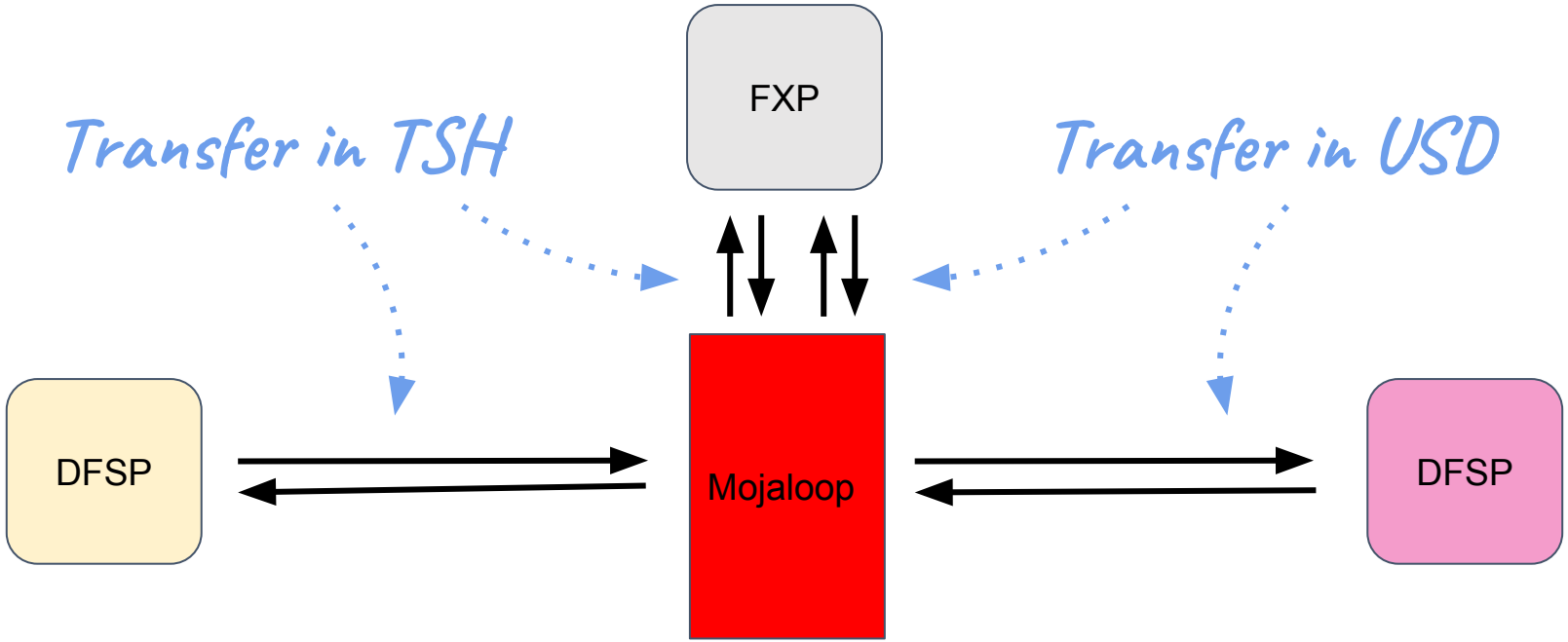


Yellow Mobile
DFSP in Tanzania

Super FX
FX provider

Pink Mobile
DFSP in Tanzania

Two transfers



Yellow Mobile
DFSP in Tanzania

Red Moja
Both USD and TSH clearing accounts

Pink Mobile
DFSP in Tanzania

Community Contributions

Mock DFSP (Deprecated)

- A payee service that responds to API calls as a mock payee DFSP

Interop Switch JS

- A Javascript implementation of the interop-switch for proxying lookup and quote API calls between DFSPs with routing

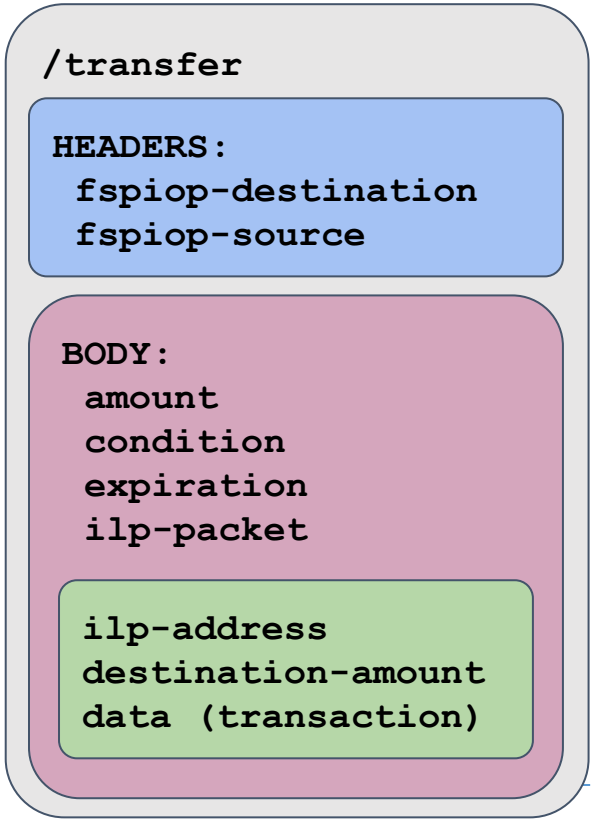
Visualizations (Not yet public)

- End-to-end transaction visualization dashboard

Proposed API Changes

- Align with changes to ILP introduced between ILPv1 and ILPv4
- `/transfer` API maps directly to ILPv4 packet headers
  a. ILP Address (in headers as `FspId`)
  b. Transfer Amount (in `transfer` object)
  c. Condition (in `transfer` object)
  d. Expiry (in `transfer` object)
  e. Data (`transfer.transaction` is the end-to-end ILPv4 payload)

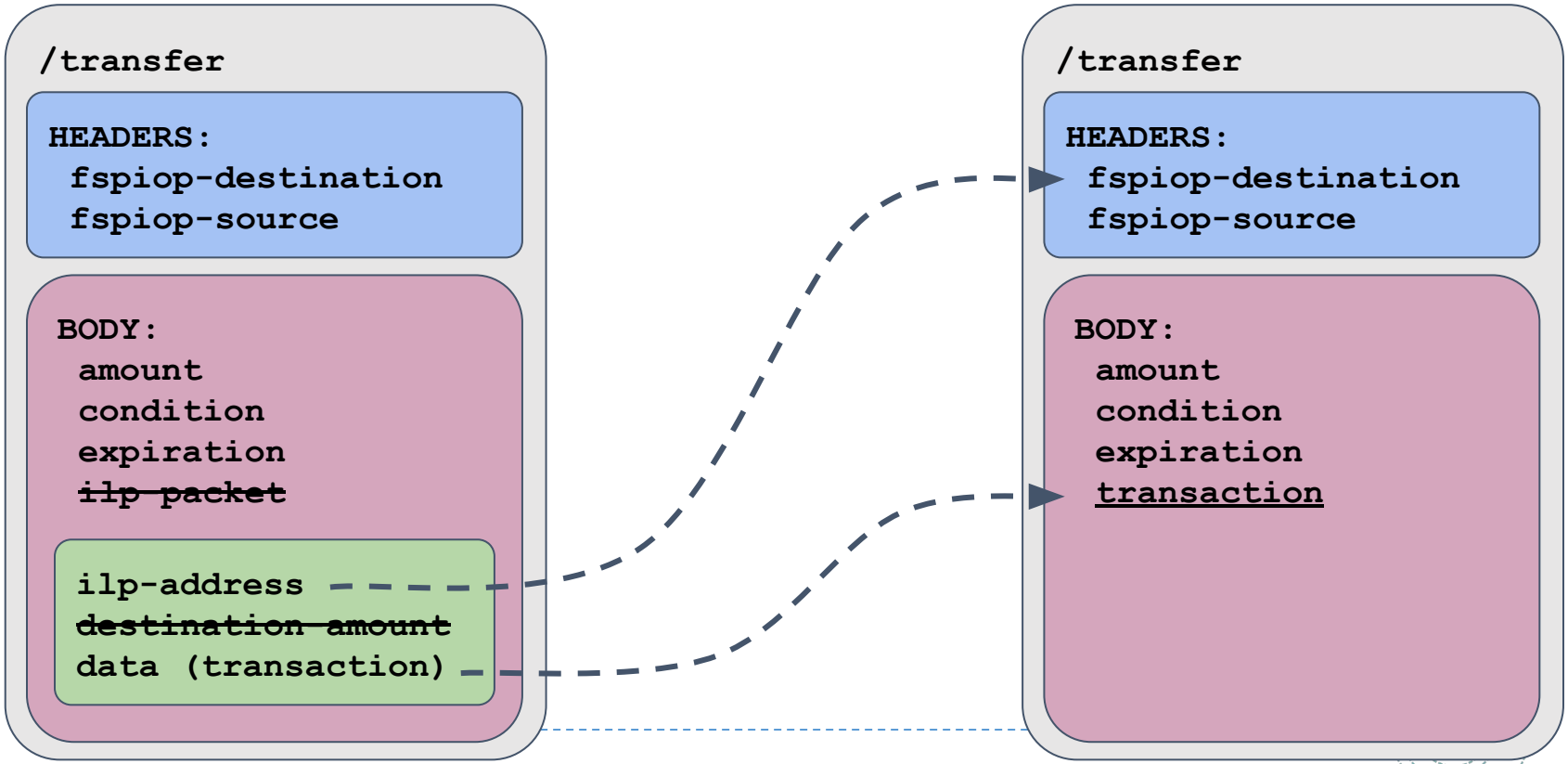
Proposed API Changes



Current API

- OER encoded ILP packet embedded in transfer
- ILP Address in ILP packet headers
- Transaction in ILP packet payload

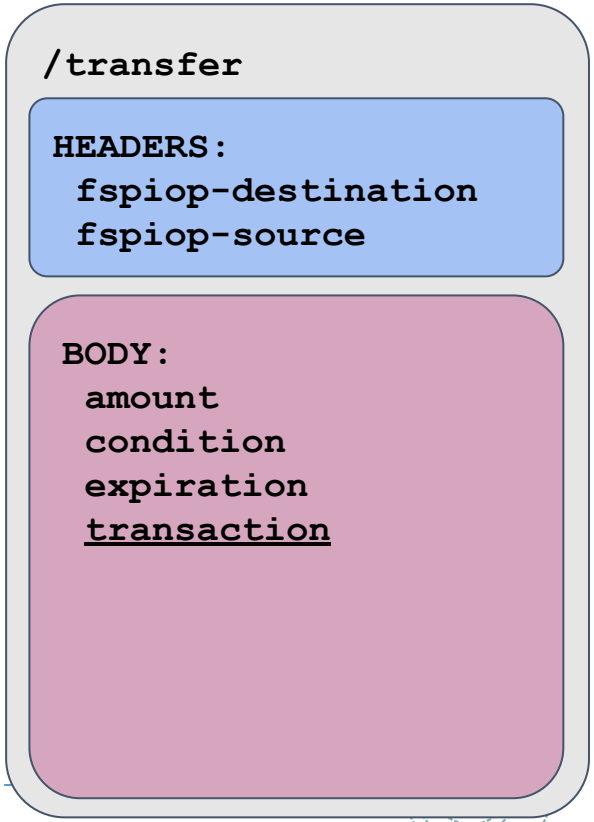
Proposed API Changes



Proposed API Changes

New API

- `ilp-address` is now in the transfer headers as `FspId`
- Elevate `transaction` object to be a field in the `transfer` object
- `destination-amount` is no longer required
- `ilp-packet` can be removed from transfer object



Proposed API Changes (impact)

- Simpler logic for derivation of `condition` and `fulfillment`
  - `fulfillment = SHA256-HMAC(transaction)`
  - `condition = SHA256(fulfillment)`
- However, logic for verification of fulfillment is **unchanged**
- Correlation between `/quote` and `/transaction` through `transfer.transaction.transactionId` and `transfer.transaction.quoteId`
- No OER encoding

Next Steps

- Model both fixed send and fixed receive amounts
- Dynamic routing and route data exchange between participants
- Include regulatory data exchange in the quote flow
- Model “multiple CNP” scenarios
  - Sending multiple quotes
  - Quote selection strategies
  - Routing transfers to follow best quote
- Model the “cross-currency, single Mojaloop” scenario
  - Deploy FX provider
  - Configure Mojaloop with multiple currencies

Questions

- Is this what you expected to see?
- Should we proceed with the POC and are we on the right track?
- How can we work more closely with the rest of the community?
- Are there use cases we should prioritise?
- Are the API changes ready to propose to the CCB?