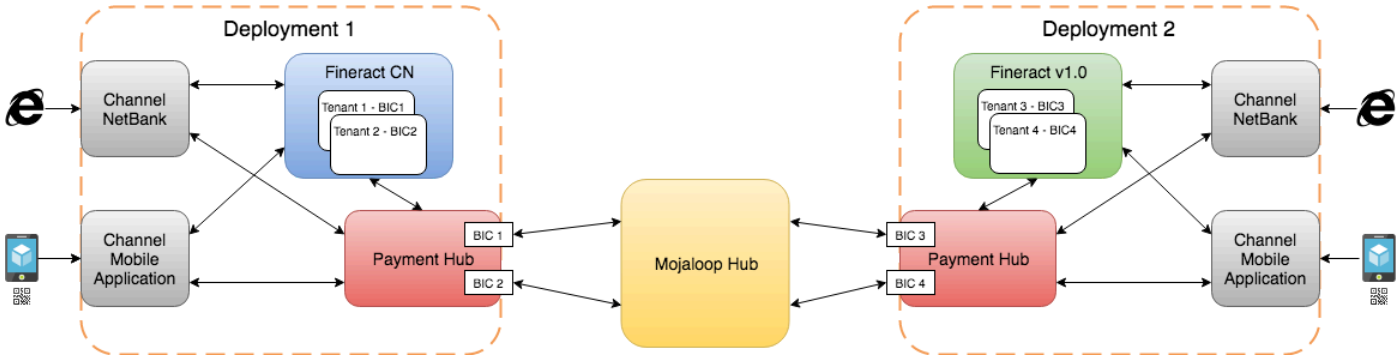
Functional and Technical Specification
of
integrating
Fineract with Mojaloop

DPC

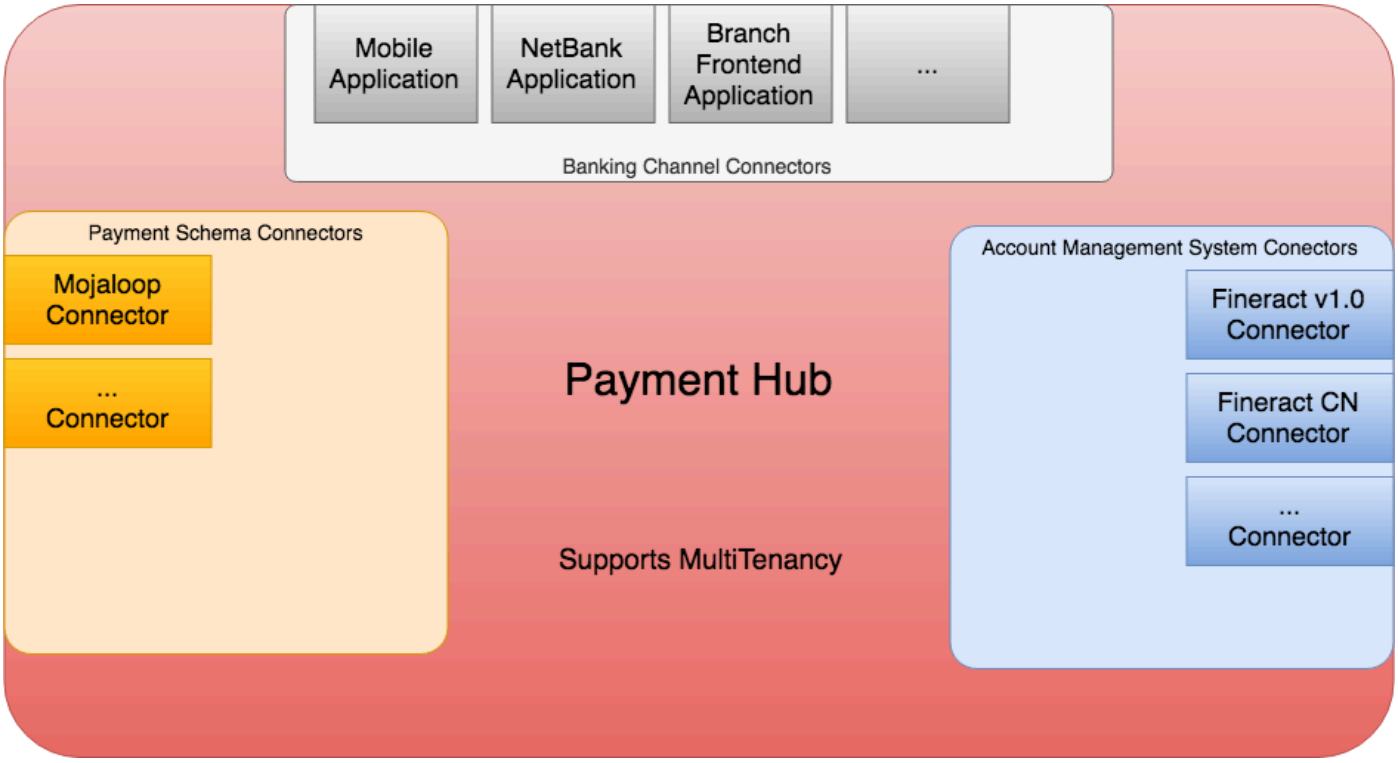
2018.

1 Architecture

The “FSP Backend” components will be implemented using the different versions of Fineract.
There will be a **Payment Hub** implemented, to **manage the communication via the Mojaloop Hub** system.
The backend of **Fineract v1.0** and **Fineract CN** might be modified to be able to **handle the required functionality**, triggered by the the Payment Hub. All communication between the Fineract and the Payment Hub will be implemented via synchronous REST API in this demo version, authenticated with a technical user. Later it can be extended to support different communication standards.
All communication between the Payment Hub and the Mojaloop will be implemented via asynchronous **REST API**.
To prevent dependency of the implementation of the banking channels, every notification will fall back to be successful scenario - means will consider every request as confirmed after a timeout.



Architecture


Mobile Application
NetBank Application
Branch Frontend Application
...
Banking Channel Connectors
Payment Schema Connectors
Mojaloop Connector
... Connector
Payment Hub
Supports MultiTenancy
Account Management System Conectors
Fineract v1.0 Connector
Fineract CN Connector
... Connector

Payment Hub

2 Use cases

2.1 A user sends an amount to a Merchant

MPH-14
Merchant Payment

Highest priority. 60-80% of the transactions are these kind.
The payee is identified by a registered primary identifier (not telephone number, nor another related attribute), which can be resolved by Mojaloop.
E.g.: a user does a purchase at a merchant. Gets the invoice, on which there is a QR code printed. The QR code contains all info of the transfer: the amount of the purchase, and the ID of the merchant. That ID can be resolved by Mojaloop. The user logs into the corresponding mobile application (so the initiator of the transaction is known), reads the QR code with a mobile application (so all other required payee info is known for the transaction from the QR code).
The payer user requests an amount to be transferred, so that amount will arrive to the merchant. This means, that because of fees or costs, it is possible, that higher amount will be withdrawn from the payer.

2.1.1 Workflow

- The **payer** initiates the **payment** using an interface of a channel (Mobile Application most probably, InternetBank, or other).
- The channel triggers the payer-side Payment Hub by the corresponding REST call - see API specification.
- The payer-side Payment Hub have to be able to instruct Fineract to lock the amount on the payer's account.
- The payer-side Payment Hub handles the sending workflow and related calls with Mojaloop and Fineract:
    - lookup of the payer and payee user via Mojaloop
    - Payment Hub maps the Mojaloop identifier to tenant+account identifier
    - Payment Hub does the authentication if necessary with Mojaloop and Fineract - demo version: just checks, if the request-sender user exists
    - quote: gathering transaction costs and fees directly from payer-side Fineract and via Mojaloop from the payee side
    - handles the transfer flow

  During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.
- The receiver-side Payment Hub handles the receiving workflow and related calls with Mojaloop and Fineract:
    - responds to account lookup requests

    - responds to costs and fees request by calling forward to Fineract
    - handles the transfer flow

  During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.
- Based on the instructions of the Payment Hubs, the connected Fineracts acts correspondingly, provide the requested data, persist the status changes, book the amount on the accounts.

2.1.2 Banking Channel related requirements

The related channel should be able to

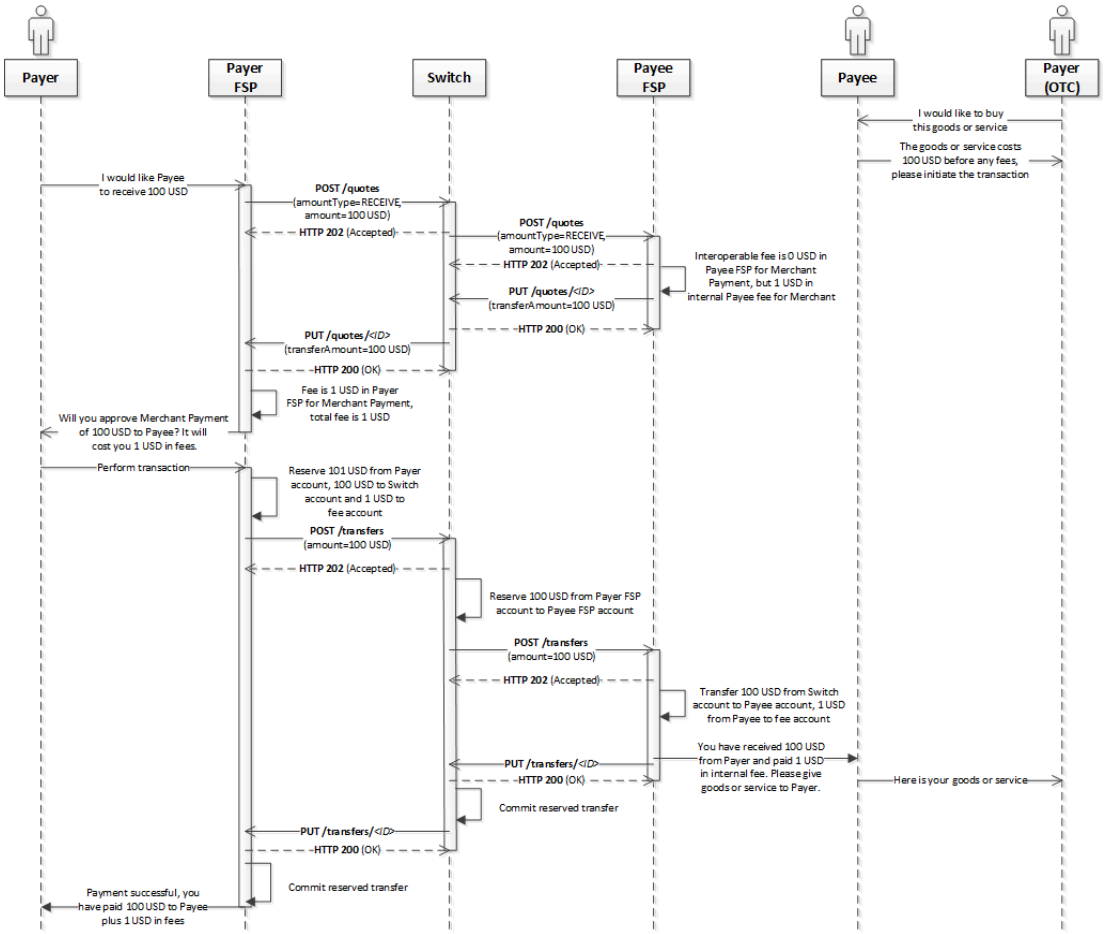
- read and decode QR code, - MPH-18
- call the corresponding REST endpoint of Payment Hub, - MPH-19
- provide a REST endpoint to be called for notification purposes, - MPH-20
- request status information about transactions via REST call to Payment Hub. - MPH-21

2.1.3 API Specification

- *5.3.6 Customer-Initiated Merchant Payment*

  To perform a Customer-Initiated Merchant Payment, set elements as follows:
    - TransactionScenario to PAYMENT
    - TransactionInitiator to PAYER
    - TransactionInitiatorType to CONSUMER
- *Payer Receive amount*
    - Receive amount should be interpreted as the amount that should be added to the Payee's account, regardless of any interoperable transaction fees. The amount excludes possible internal Payee fees added by the Payee FSP.
- *8.1 Payer Initiated Transaction*

  The Payer Initiated Transaction pattern is introduced in Generic Transaction Patterns. On a high-level, the pattern should be used whenever a Payer would like to transfer funds to another Party whom is not located in the same FSP as the Payer.



2.2 A user requests amount from another user - MPH-15 Peer-to-Peer (P2P) transfer

Medium Priority.

E.g.:

- A user asks for money from parents;

The payee user requests an amount to be transferred. Exactly that amount will be withdrawn. Fees and costs might be applied on payer and payee side. This means, that it is possible, that a reduced amount will arrive to the payee.
Rewards can also be applied.

2.2.1 Workflow

- The **payee** initiates the **payment with a payment request** using an interface of a channel (Mobile Application most probably, InternetBank, or other).
- The channel triggers the payee-side Payment Hub by a REST call, by sending in

    - the payer identifier,
    - the payee identifier and
    - the amount.
- The payee-side Payment Hub triggers the corresponding workflow, to request the amount via Mojaloop.
- The payer-side Payment Hub gets a call from Mojaloop about the payment request.
- The payer-side Payment Hub sends a message with a notification to the registered channel, about the payment request. To prevent dependency of the implementation of the banking channels, every notification will fall back to be successful.
- The payer-side channel sends confirmation to the Payment Hub.
- The payer-side Payment Hub have to be able to instruct Fineract to lock the amount on the payer's account.
- The payer-side Payment Hub handles the sending workflow and related calls with Mojaloop and Fineract:
    - lookup of the payer and payee user via Mojaloop
    - Payment Hub maps the Mojaloop identifier to tenant+account identifier
    - Payment Hub does the authentication if necessary with Mojaloop and Fineract - demo version: just checks, if the request-sender user exists
    - quote: gathering transaction costs and fees directly from payer-side Fineract and via Mojaloop from the payee side
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- The receiver-side Payment Hub handles the receiving workflow and related calls with Mojaloop and Fineract:
    - responds to account lookup requests
    - responds to costs and fees request by calling forward to Fineract
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- Based on the instructions of the Payment Hubs, the connected Fineracts acts correspondingly, provide the requested data, persist the status changes, book the amount on the accounts.

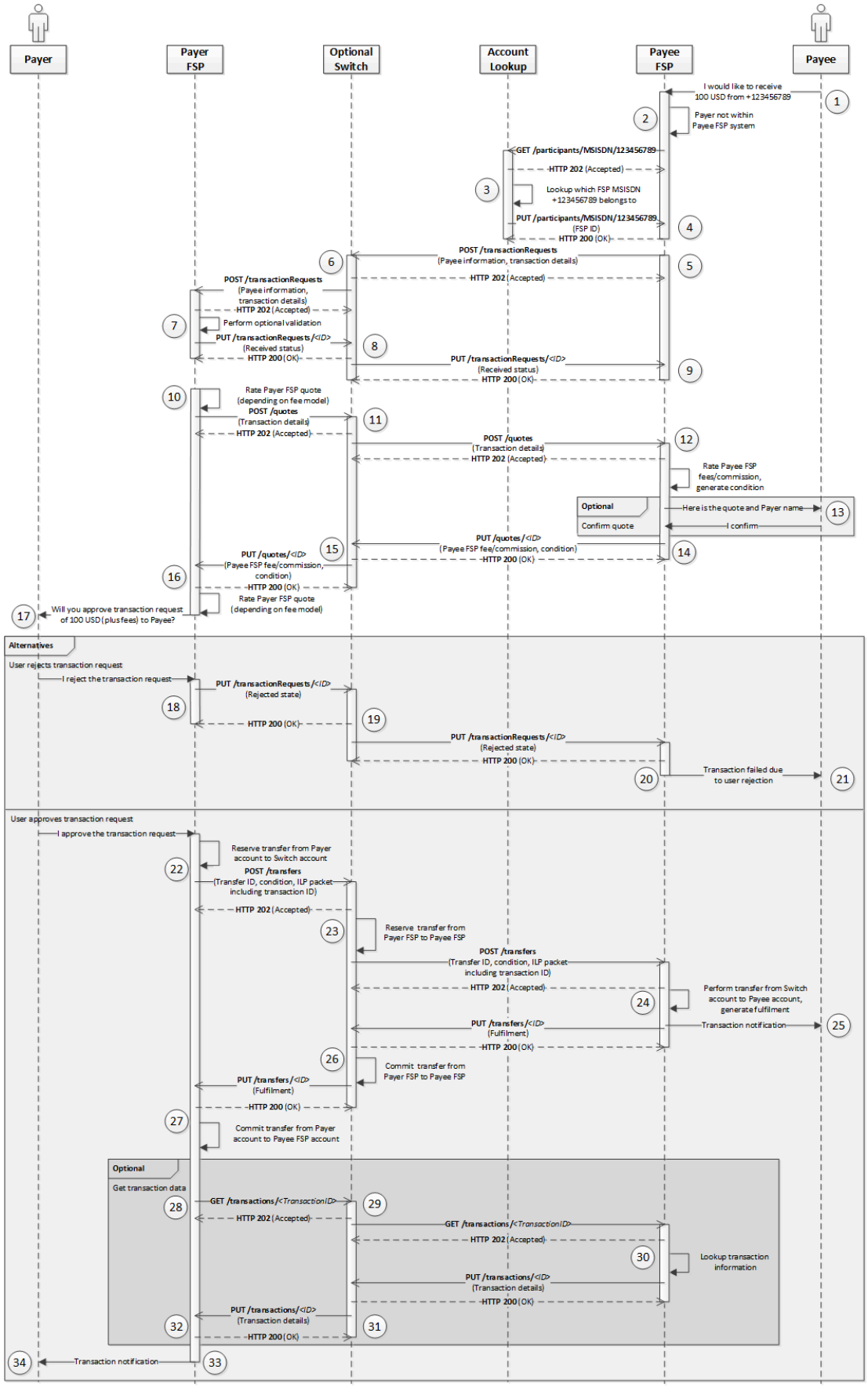
2.2.2 Banking Channel related requirements

The related channel should be able to

- handle secondary identifier like phone number or email address and send it to the Payment Hub,
- GUI for initiating the transaction - MPH-23
- call the corresponding REST endpoint of Payment Hub, - MPH-24
- provide a REST endpoint to be called for notification purposes, - MPH-20
- request status information about transactions via REST call to Payment Hub. - MPH-21

2.2.3 API Specification

- *5.3.1 P2P Transfer*
  To perform a P2P Transfer, set elements as follows:
  - TransactionScenario to TRANSFER
  - TransactionInitiator to PAYEE
  - TransactionInitiatorType to CONSUMER
- *Payer Send amount*.
  - Send amount should be interpreted as the actual amount that should be deducted from the Payer's account, including any fees.
- *8.2 Payee Initiated Transaction*
  The Payee Initiated Transaction pattern is introduced in Generic Transaction Patterns. On a high-level, the pattern should be used whenever a Payee would like to request that Payer transfer funds to a Payee. The Payer and the Payee are assumed to be in different FSPs, and the approval of the transaction is performed in the Payer FSP. If the transaction information and approval occur on a Payee device instead, use the related Payee Initiated Transaction using OTP pattern described in Section 8.3 instead.


Payer
Payer FSP
Optional Switch
Account Lookup
Payee FSP
Payee
I would like to receive 100 USD from +123456789
1
2
Payer not within Payee FSP system
GET /participants/MSISDN/123456789
HTTP 202 (Accepted)
3
Lookup which FSP MSISDN +123456789 belongs to
PUT /participants/MSISDN/123456789 (FSP ID)
4
HTTP 200 (OK)
6
POST /transactionRequests (Payee information, transaction details)
5
HTTP 202 (Accepted)
POST /transactionRequests (Payee information, transaction details)
HTTP 202 (Accepted)
7
Perform optional validation
PUT /transactionRequests/<ID> (Received status)
8
HTTP 200 (OK)
PUT /transactionRequests/<ID> (Received status)
9
HTTP 200 (OK)
10
Rate Payer FSP quote (depending on fee model)
POST /quotes (Transaction details)
11
HTTP 202 (Accepted)
POST /quotes (Transaction details)
12
HTTP 202 (Accepted)
Rate Payee FSP fees/commission, generate condition
Optional
Confirm quote
Here is the quote and Payer name
13
I confirm
PUT /quotes/<ID> (Payee FSP fee/commission, condition)
15
HTTP 200 (OK)
14
PUT /quotes/<ID> (Payee FSP fee/commission, condition)
16
HTTP 200 (OK)
Rate Payer FSP quote (depending on fee model)
17
Will you approve transaction request of 100 USD (plus fees) to Payee?
Alternatives
User rejects transaction request
I reject the transaction request
PUT /transactionRequests/<ID> (Rejected state)
18
HTTP 200 (OK)
19
PUT /transactionRequests/<ID> (Rejected state)
HTTP 200 (OK)
20
Transaction failed due to user rejection
21
User approves transaction request
I approve the transaction request
Reserve transfer from Payer account to Switch account
22
POST /transfers (Transfer ID, condition, ILP packet including transaction ID)
HTTP 202 (Accepted)
23
Reserve transfer from Payer FSP to Payee FSP
POST /transfers (Transfer ID, condition, ILP packet including transaction ID)
HTTP 202 (Accepted)
24
Perform transfer from Switch account to Payee account, generate fulfilment
PUT /transfers/<ID> (Fulfilment)
Transaction notification
25
HTTP 200 (OK)
26
Commit transfer from Payer FSP to Payee FSP
PUT /transfers/<ID> (Fulfilment)
HTTP 200 (OK)
27
Commit transfer from Payer account to Payee FSP account
Optional
Get transaction data
28
GET /transactions/<TransactionID>
HTTP 202 (Accepted)
29
GET /transactions/<TransactionID>
HTTP 202 (Accepted)
30
Lookup transaction information
PUT /transactions/<ID> (Transaction details)
HTTP 200 (OK)
PUT /transactions/<ID> (Transaction details)
32
HTTP 200 (OK)
31
34
Transaction notification
33

2.3 A user sends amount to another user - MPH-16
Peer-to-Peer (P2P) transfer

2.3.1 Workflow

- The **payer** initiates the **payment** using an interface of a channel (Mobile Application most probably, InternetBank, or other).
- The channel triggers the payer-side Payment Hub by the corresponding REST call - see API specification.
- The payer-side Payment Hub have to be able to instruct Fineract to lock the amount on the payer's account.
- The payer-side Payment Hub handles the sending workflow and related calls with Mojaloop and Fineract:
    - lookup of the payer and payee user via Mojaloop
    - Payment Hub maps the Mojaloop identifier to tenant+account identifier
    - Payment Hub does the authentication if necessary with Mojaloop and Fineract - demo version: just checks, if the request-sender user exists
    - quote: gathering transaction costs and fees directly from payer-side Fineract and via Mojaloop from the payee side
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- The receiver-side Payment Hub handles the receiving workflow and related calls with Mojaloop and Fineract:
    - responds to account lookup requests
    - responds to costs and fees request by calling forward to Fineract
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- Based on the instructions of the Payment Hubs, the connected Fineracts acts correspondingly, provide the requested data, persist the status changes, book the amount on the accounts.

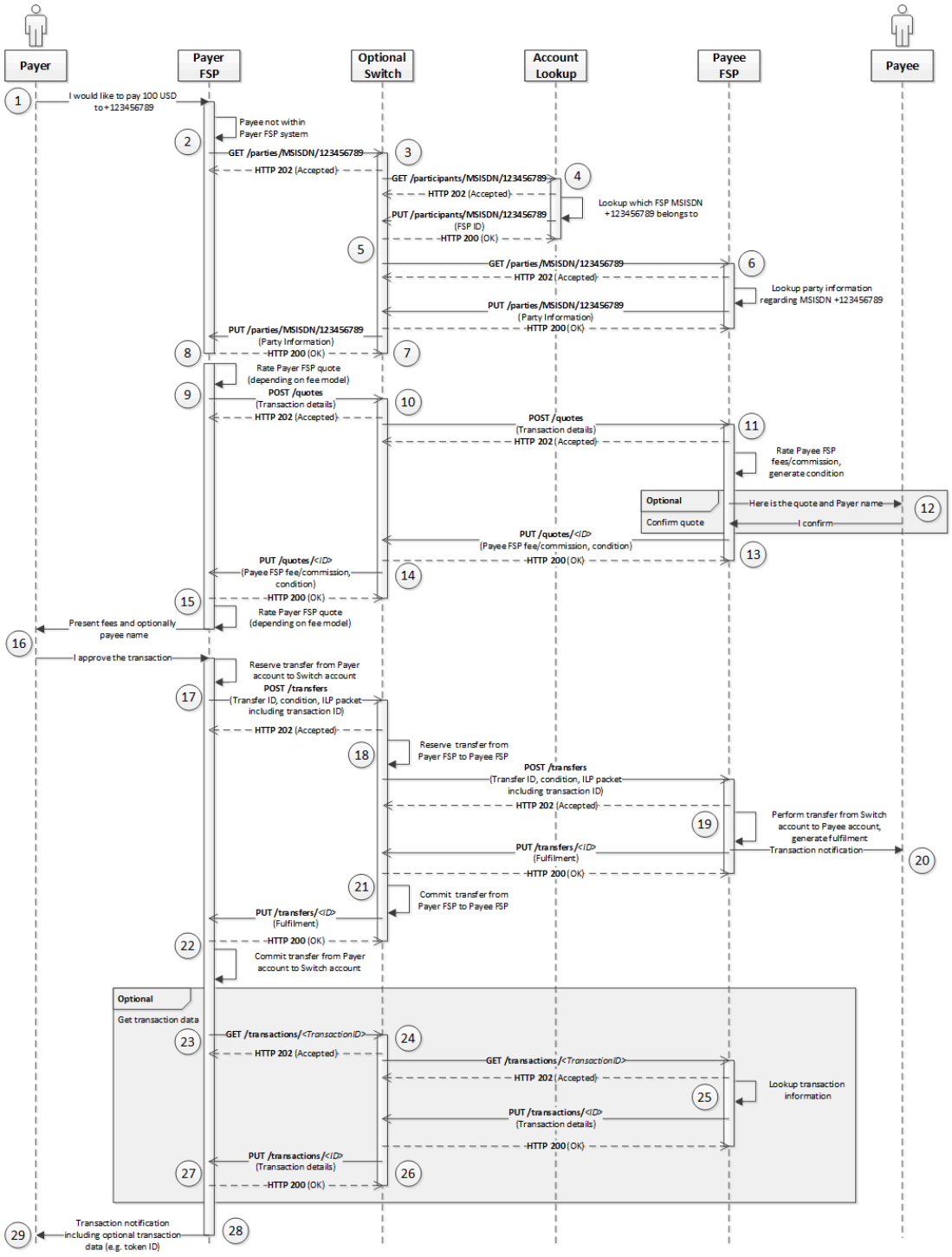
2.3.2 Banking Channel related requirements

The related channel should be able to

- handle secondary identifier like phone number or email address and send it to the Payment Hub,
- GUI for initiating the transaction - MPH-25
- call the corresponding REST endpoint of Payment Hub, - MPH-26
- provide a REST endpoint to be called for notification purposes, - MPH-20
- request status information about transactions via REST call to Payment Hub. - MPH-21

2.3.3 API Specification

- *5.3.1 P2P Transfer*
  To perform a P2P Transfer, set elements as follows:
  - TransactionScenario to TRANSFER
  - TransactionInitiator to PAYER
  - TransactionInitiatorType to CONSUMER
- *Payer Send amount*.
  - Send amount should be interpreted as the actual amount that should be deducted from the Payer's account, including any fees.
- *8.1 Payer Initiated Transaction*
  The Payer Initiated Transaction pattern is introduced in Generic Transaction Patterns. On a high-level, the pattern should be used whenever a Payer would like to transfer funds to another Party whom is not located in the same FSP as the Payer.



2.4 A user wants to send different amounts to different user accounts in a batch - Batch Salary Disbursement - MPH-17

Lowest Priority.

Note: Apache CN Payroll Service might help - check the functionality.

2.4.1 Workflow

- The **payer** initiates the **payment** using an interface of a channel (Mobile Application most probably, InternetBank, or other): uploads a csv file that defines the requested transfers.
- The channel triggers the payer-side Payment Hub by the corresponding REST call - see API specification.
- The payer-side Payment Hub have to be able to instruct Fineract to lock the amount on the payer's account.
- The payer-side Payment Hub handles the sending workflow and related calls with Mojaloop and Fineract:
    - lookup of the payer and payee user via Mojaloop
    - Payment Hub maps the Mojaloop identifier to tenant+account identifier
    - Payment Hub does the authentication if necessary with Mojaloop and Fineract - demo version: just checks, if the request-sender user exists
    - quote: gathering transaction costs and fees directly from payer-side Fineract and via Mojaloop from the payee side
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- The receiver-side Payment Hub handles the receiving workflow and related calls with Mojaloop and Fineract:
    - responds to account lookup requests
    - responds to costs and fees request by calling forward to Fineract
    - handles the transfer flow

During the flow the Payment Hub instructs Fineract to provide necessary information, persist data and status changes where it is required.

- Based on the instructions of the Payment Hubs, the connected Fineracts acts correspondingly, provide the requested data, persist the status changes, book the amount on the accounts.
- The requestor gets back a csv file with the results of the transactions: were they successful, or error occurred.

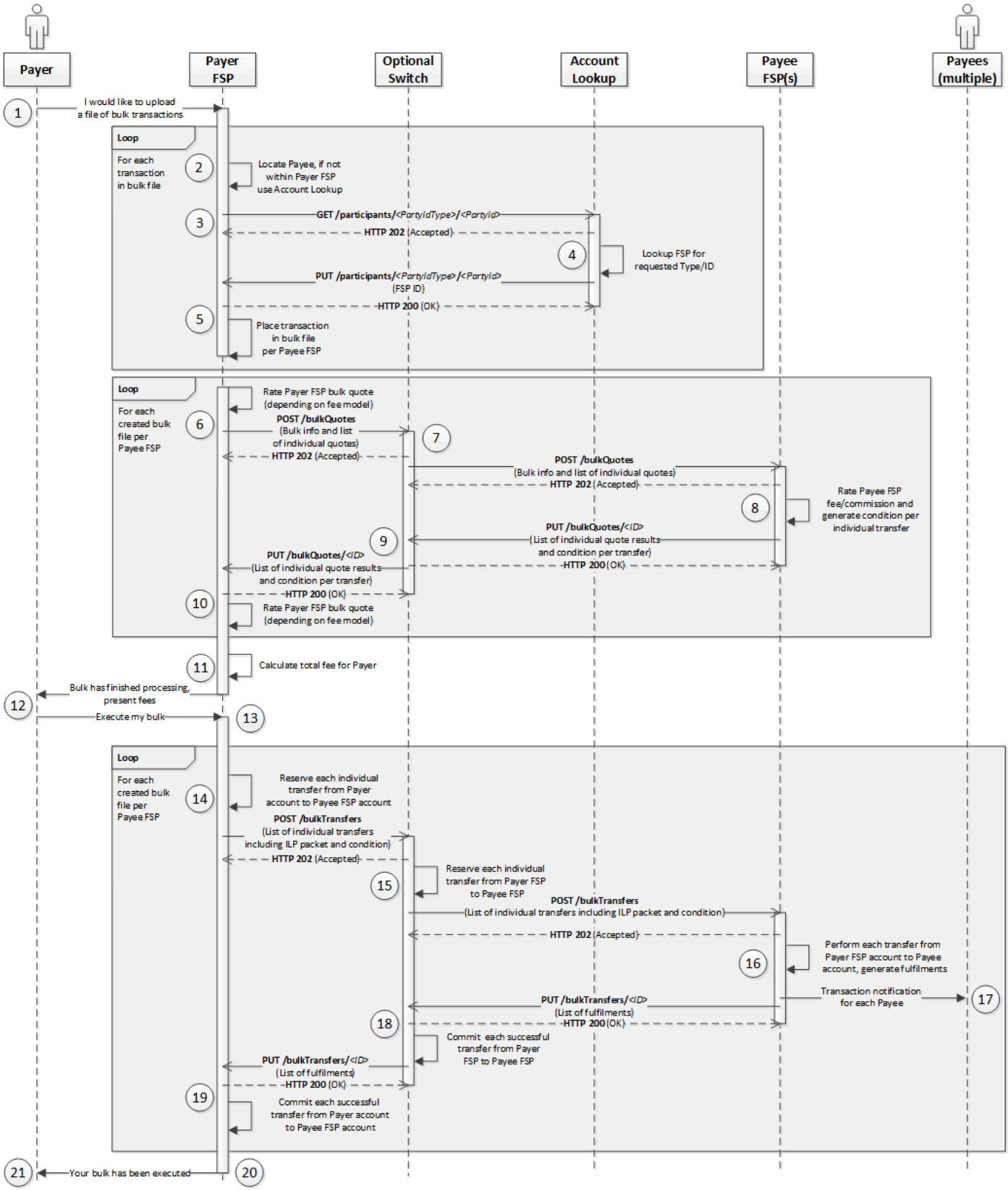
2.4.2 Banking Channel related requirements

The related channel should be able to

- GUI for uploading a csv file that defines the requested transfers, - MPH-27
- send a csv file via REST that defines the requested transfers, - MPH-28
- request status information about transactions via REST call to Payment Hub, - MPH-21
- provide REST endpoint to be called with the response csv file, - MPH-29
- the response csv with the result of the transactions should be downloadable on GUI. - MPH-30

2.4.3 API Specification

- *5.3.1 P2P Transfer*
  To perform a P2P Transfer, set elements as follows:
  - TransactionScenario to TRANSFER
  - TransactionInitiator to PAYER
  - TransactionInitiatorType to CONSUMER.
- *8.4 Bulk Transactions*
  The Bulk Transaction pattern is introduced in Generic Transaction Patterns. On a high-level, the pattern is used whenever a Payer would like to transfer funds to multiple Payees using one single transaction. The Payees can be in different FSPs.


Payer
Payer FSP
Optional Switch
Account Lookup
Payee FSP(s)
Payees (multiple)
1
I would like to upload a file of bulk transactions
Loop
For each transaction in bulk file
2
Locate Payee, if not within Payer FSP use Account Lookup
3
GET /participants/<PartyIdType>/<PartyId>
HTTP 202 (Accepted)
4
Lookup FSP for requested Type/ID
PUT /participants/<PartyIdType>/<PartyId>
(FSP ID)
HTTP 200 (OK)
5
Place transaction in bulk file per Payee FSP
Loop
For each created bulk file per Payee FSP
Rate Payer FSP bulk quote (depending on fee model)
6
POST /bulkQuotes
(Bulk info and list of individual quotes)
HTTP 202 (Accepted)
7
POST /bulkQuotes
(Bulk info and list of individual quotes)
HTTP 202 (Accepted)
8
Rate Payee FSP fee/commission and generate condition per individual transfer
9
PUT /bulkQuotes/<ID>
(List of individual quote results and condition per transfer)
HTTP 200 (OK)
PUT /bulkQuotes/<ID>
(List of individual quote results and condition per transfer)
HTTP 200 (OK)
10
Rate Payer FSP bulk quote (depending on fee model)
11
Calculate total fee for Payer
12
Bulk has finished processing, present fees
Execute my bulk
13
Loop
For each created bulk file per Payee FSP
14
Reserve each individual transfer from Payer account to Payee FSP account
POST /bulkTransfers
(List of individual transfers including ILP packet and condition)
HTTP 202 (Accepted)
15
Reserve each individual transfer from Payer FSP to Payee FSP
POST /bulkTransfers
(List of individual transfers including ILP packet and condition)
HTTP 202 (Accepted)
16
Perform each transfer from Payer FSP account to Payee account, generate fulfilments
Transaction notification for each Payee
17
PUT /bulkTransfers/<ID>
(List of fulfilments)
HTTP 200 (OK)
18
Commit each successful transfer from Payer FSP to Payee FSP
PUT /bulkTransfers/<ID>
(List of fulfilments)
HTTP 200 (OK)
19
Commit each successful transfer from Payer account to Payee FSP account
21
Your bulk has been executed
20

3 Epics

3.1 Fineract v1.0 install and config

3.2 Fineract CN install and config

3.3 Setup Mojaloop infrastructure with test data

MPH-33

3.4 AWS/Azure Cloud infrastructure for Fineract v1.0

MPH-31

3.5 AWS/Azure Cloud infrastructure for Fineract CN

MPH-32

3.6 Gap Analysis for Fineract v1.0

MPH-34

3.7 Gap Analysis for Fineract CN

MPH-35

3.8 Design and implement Payment Hub

MPH-1

- Payment Hub wants to do Authentication when triggering Fineract versions - using technical user - MPH-5, MPH-6
- REST API for receiving requests from the channels - MPH-12
    - merchant payment
    - sender-initiated transfer
    - receiver-initiated transfer
    - bulk transfer
    - confirmation of actions (E.g. when a payer netbank user confirms a payment request)

- REST API for sending requests for channels (E.g. when a netbank user has to be notified about confirming a payment request) - MPH-12
    - Payment Hub wants to notify the user via REST call to configured REST endpoints to transfer the notifications
- REST API for initiating requests to and receiving requests from Fineract (CN or v1.0) - MPH-4, MPH-3
- REST API for initiating requests to and receiving requests from Mojaloop Hub - MPH-2
- Manage the 2 main workflows
    - sender-user initiated workflow
    - receiver-user initiated workflow

  Other sub-scenarios are just matter of parameterisation.
- Manage bulk transfer
    - do transactions based on input file, which can be transferred via REST API.
    - provide response in an output file, which can be responded via REST API.

3.9 Documentation analysis, initial discussions

3.10 Fineract v1.0 - Payer side related features - MPH-9

3.11 Fineract v1.0 - Payee side related features - MPH-9

3.12 Fineract CN - Payer side related features - MPH-10

3.13 Fineract CN - Payee side related features - MPH-10

3.14 Error Handling

3.15 Data Validation - API Requests

3.16 GUI - not for DPC TBD - MPH-13

3.17 Send Messages to the Channel backend

E.g. when a netbank user has to be notified about confirming a payment request

3.18 Receive Messages from the Channel backend

3.19 Banking Channel upgrades for handling the use cases - not for DPC

- Channel GUI
- Channel backend should send messages via REST API
- Channel backend should receive messages via REST API

3.19.1 Payment from user to merchant

3.19.2 Payment from user to user

3.19.3 Payment request from user to user

3.19.4 Batch payment from user to user

3.19.5 Lookup primary identifier based on secondary identifier

3.19.6 Confirmations

- Confirmation of payment request

3.19.7 Notifications

- Notification about successful transaction
- Notification about error

3.20 Fee distribution on receiver side

If there are more kind of fees defined for a transaction, than those have to be booked to the corresponding accounts. Accounts should be able to be configured, and the configured fees should be handled accordingly.

4 To be implemented on later phase

Including but not limited to:

- API versioning (3.3 in pdf), Accept Header version

- Registration for lookup service - Lookup Participant Information (LPI), Lookup Party Information. Registration as trusted. Synchronisation with this service.
- Trusted payee/payer registration and maintenance.
- Confirmation of non trusted payee. No push notification will be implemented.
- Confirmations of fees. No push notification will be implemented.
- Refund is specified in API documentation. Refund is also out of scope. (refund is used instead of reversal).
- Geolocation handling: lat, lon - GUI function. The server will provide possibility to store.
- Currency validation (allowed currencies) and exponent - CONFIRMED
- Fee distribution on receiver side - IN SCOPE!
- Fee collection on sender side when it is not collected from the sender (e.g. would be paid by the bank) - NOT in scope, CONFIRMED
- Try to minimise error handling
- ILP: we can skip.
- Enhancement of Fineract back-office screens could be part of a next phase, in this scope triggering of these scenarios will be supported by the banking channel interfaces.

I write here sg to test versioning

5 Questions

- Is the API doc v1.0 the valid documentation? - yes
- Is there tester Postman collection available? We found only a limited version. - We will get a newer version if exists.
- Is everything really async? Even the GET calls? - yes
- Is LPI implemented in Mojaloop? Most probably yes, based on the architecture figure.
- ILP: we can skip. No hashing, etc.
- Which confirmations/accepts are must have? E.g.: fees after quote, etc
- Is the trusted payer/payee maintenance mandatory?
- *6.7.1.6 Optional Additional Clearing Check*
- There are some workflows in the API documentation, where some steps are omitted, like: quote, authorisation, requestTransaction.
    - *Payer Initiated Transaction*
    - *8.2 Payee Initiated Transaction*
    - *8.3 Payee Initiated Transaction*
    - *8.4 Bulk Transactions*
    - *10.2 End-to-End Flow*
- There are some workflows in the API documentation, where the switch (=? Mojaloop) is optional. Why is that?
    - *6.3.1 Service Details*
    - *6.7.1.5 Client Receiving Expired Transfer*
    - *6.10.2.2 POST /bulkTransfers*

- Administration GUI in Fineract? Or init REST or an init script is enough? - NO GUI UPGRADE IN DEMO. Init scripts will be implemented.
    - create customers
    - define salary batches
    - ??
- Non-Disclosing and Disclosing fees should be supported?
  Two different modes for quoting between FSPs are supported in the API: Non-disclosing of fees and Disclosing of fees.
  The Non-Disclosing of fees mode should be the standard supported way of quoting in most schemes.
    - Non-Disclosing of fees should be used when either the Payer FSP does not want to show the Payee FSP its fee structure, or when the Payer FSP would like to have more control of the fees paid by the Payer after quoting has been performed (the latter is only applicable for Receive amount; see next bullet list).
    - Disclosing of fees can be used for use cases in which the Payee FSP wants to subsidize the transaction in some use cases; for example, Cash-In at another FSP's agent.
- Currency: "Inclusia", currency: anything that is in Fineract, let's choose Rupee. Check: 2 decimals, is it true, that the API does not support decimals?
- QR standard? Don't know. Let's collect some, and discuss it with James and Ed. Mifos devs have developed sg QR code reading - but probably that is not a mainstream version.