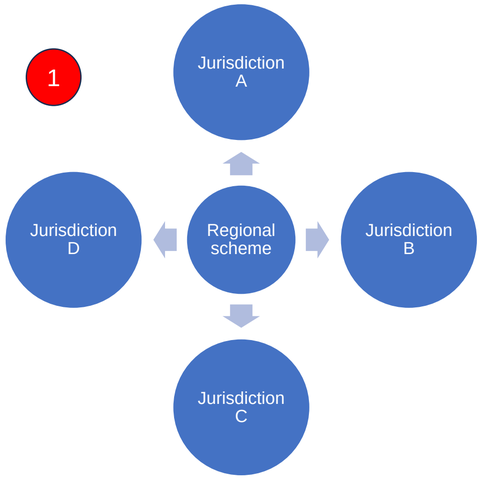
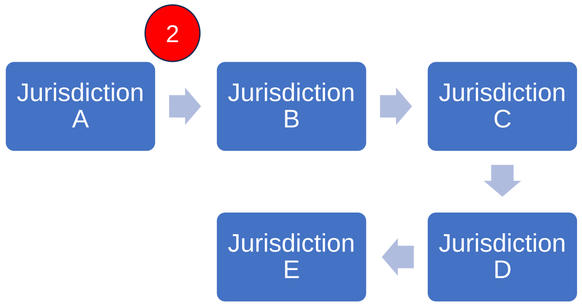
mojaloop

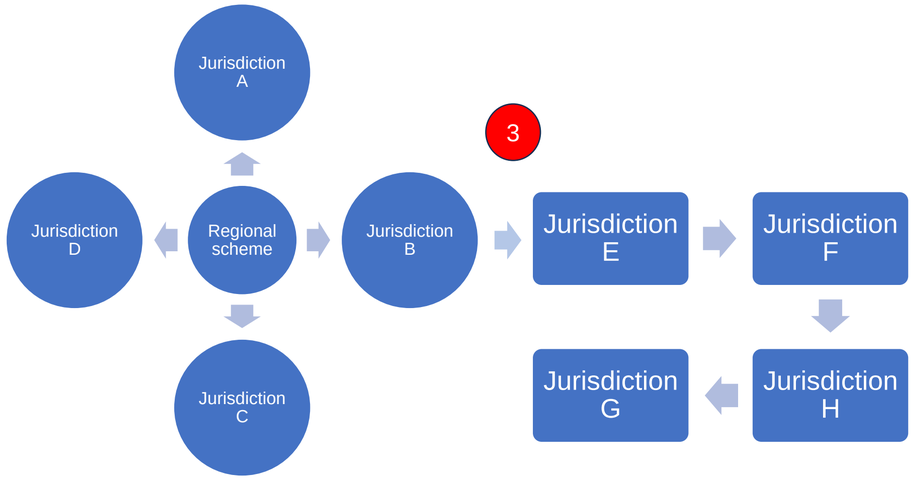
Extending Mojaloop to support a multi-scheme ecosystem

What are we trying to do?

- Support payments between customers of DFSPs which belong to different payment schemes
  - “Payment schemes” means: IIPS payment schemes
- Support any number of schemes
- Support any topography of schemes. For instance:

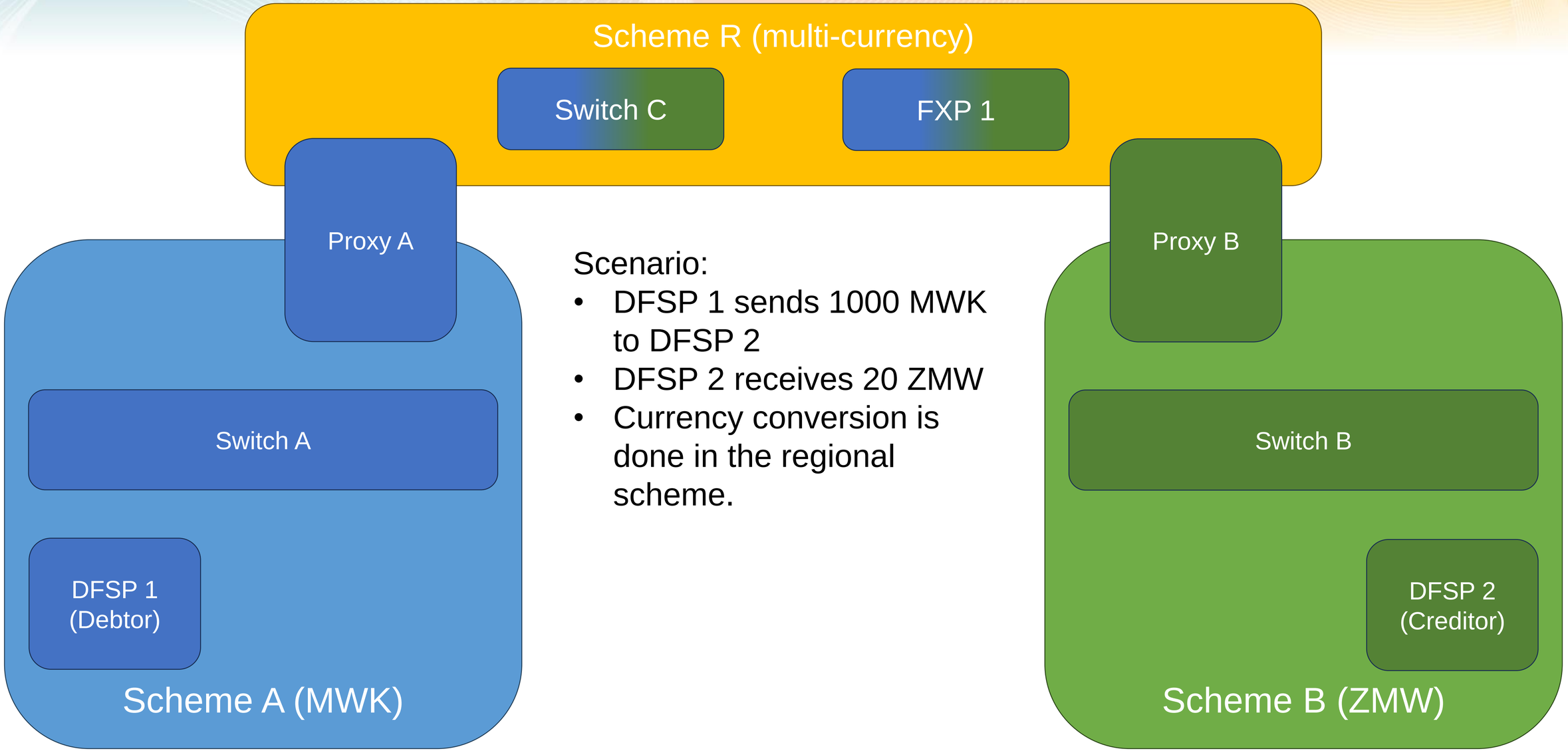






- Be invisible to participants
- Allow schemes to operate in single-scheme or multi-scheme environments without modification.
- Retain all the features of a single-scheme Mojaloop system

An example architecture



Scenario:

- DFSP 1 sends 1000 MWK to DFSP 2
- DFSP 2 receives 20 ZMW
- Currency conversion is done in the regional scheme.

What problems do we need to solve?

- How do we identify beneficiaries when they are in different countries?
- How do we know where to route a message to, if it is intended for a DFSP in another scheme?
- How do we ensure that obligations inside all schemes always balance with each other?
- How can we manage time-outs reliably?
- How are we to manage the requirements for pre-funding when a payment is passing through multiple schemes?
- How do we ensure that payments can be tracked across all the schemes through which they pass?

Beneficiary identification

Identifying participants across schemes

The situation today:

- Participants maintain their association with an identifier manually, using the **/participants** endpoint.
  - This may not be true for all oracle types.
- If an identifier is not found in an oracle, then the payment cannot be made.

In a multi-scheme environment, what are the alternatives?

- Retain the existing system
  - Forward new entries and changes to the oracles of all other schemes.
- Move to an event-driven system.

An event-driven system for address resolution

- Participants do not need to register identifiers for their customers.
- When an oracle can’t match an identifier requested by a DFSP:
  - It asks all the DFSPs it knows about whether they recognise that identifier.
  - If one does, it responds with a positive identification.
  - The oracle caches the response.
  - Next time a DFSP asks about that identifier, the oracle has an identification: it simply forwards the request to the DFSP.
  - If the customer has moved their account: the DFSP rejects the request, and the round-robin starts again.

Pros and cons

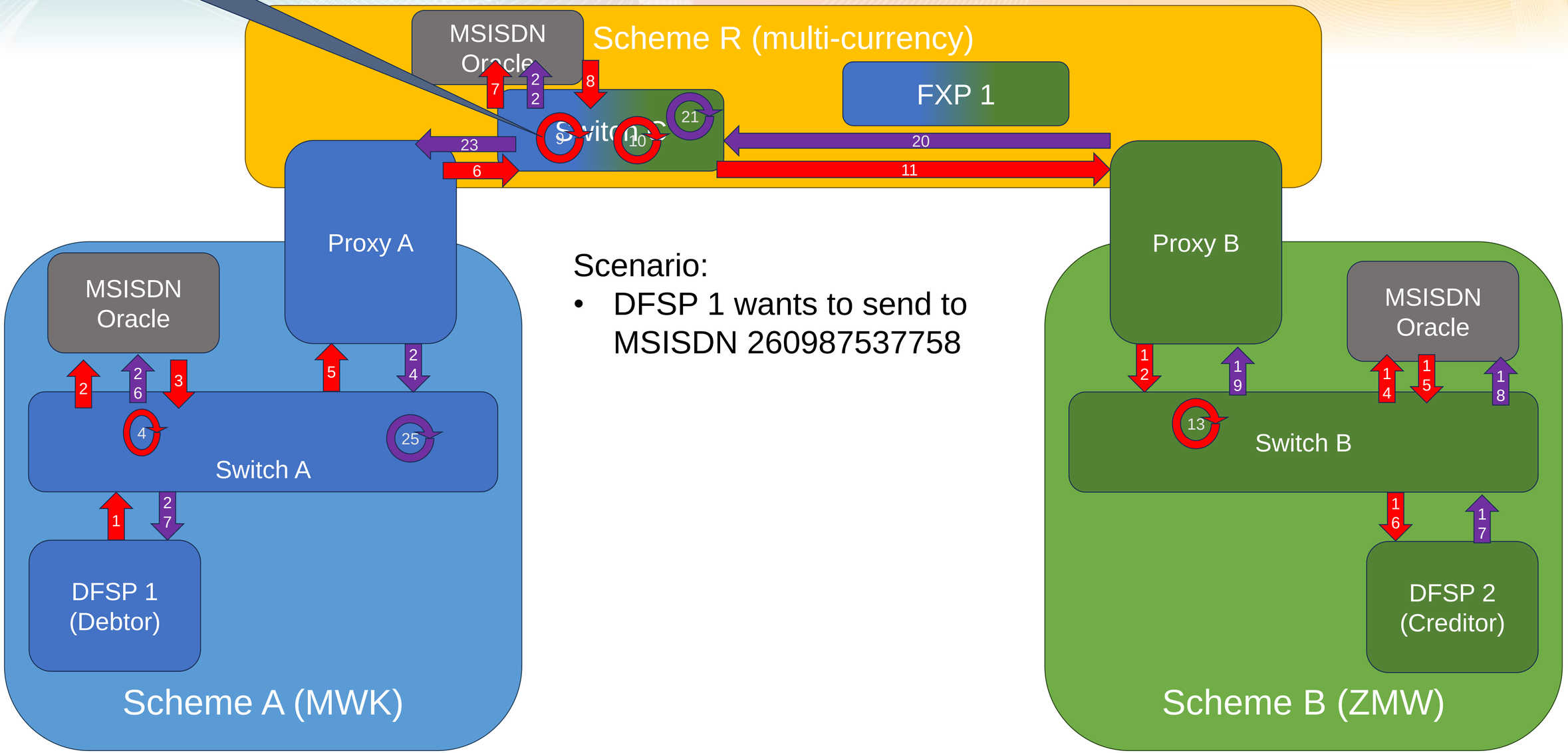
- This is expensive compared with the existing system the first time a customer is a beneficiary.
- Subsequently, it is equivalent to the existing system.
- No changes are required at the DFSP.
  - It simply responds to **GET /parties** requests in the way it does now.
- It works unmodified for oracles which connect to third party systems like PathFinder.
  - In fact, it is more efficient than the existing system for these, since results are cached on return.
- Like the existing system, it needs a strategy for coping with duplicates.
  - Cross-scheme duplicates may complicate this problem…
- We could make the system more efficient by caching the DFSP/identifier information on other calls (such as **POST /quotes** or **POST /transactionRequests**.)

Implementation

- Implementing in the ALS
  - Simple to do.
- All oracles in a scheme would work in the same way.
- Individual oracles would not need to be modified.
- Address resolution methods would be configurable
  - Existing schemes would continue to work
    - They are configured to use the existing system.

Discovery, On-demand style

The switch knows per this step that DFSP1 (the originator) is accessed via Proxy A



Scenario:

- DFSP 1 wants to send to MSISDN 260987537758

Message routing

Message routing

- Q: How do we connect schemes together?
- A: Via a proxy
  - Used to be called a Cross-Network Provider

What is a proxy?

- It’s a specialised implementation of an adapter:
  - An adapter has two interfaces.
  - In most adapters: the interfaces are different, and the adapter converts content between them.
  - In our implementation, the interfaces can be the same.
- Both sides of the adapter speak (for instance) ISO 20022.
- … but each side speaks it to a different scheme.
- The proxy acts as a concentrator
  - In any given scheme, the switch can talk to the proxy.
  - Without needing to know how to route to any of the DFSPs in the other scheme(s) on the other side of it.
- So, given that there may be more than one proxy attached to a scheme:
  - We need to be able to associate a DFSP ID with the proxy that represents it.

Our requirement

- A DFSP should be able to address a message (for instance, a request for the agreement of terms) to the intended recipient…
- … without needing to know if that recipient is in the same scheme as them or not.
- The underlying architecture should manage routing the message via the proxies to the correct destination.

Our proposal

- Proxies are registered in each scheme to which they are attached.
- The switch can obtain, on demand, a list of the proxies which are attached to it.
- When any switch receives a message from a proxy:
  - It stores the association between the DFSP who sent the message and the proxy who forwarded it.
  - Now, if it needs to send a message to that DFSP, it knows which proxy to send the message to.
- This will provide a simple mechanism to manage routing between arbitrary numbers and complexities of schemes and DFSPs.
- It is based around a simple Mojaloop principle:
  - You always find out where a DFSP is before you try to communicate with them.

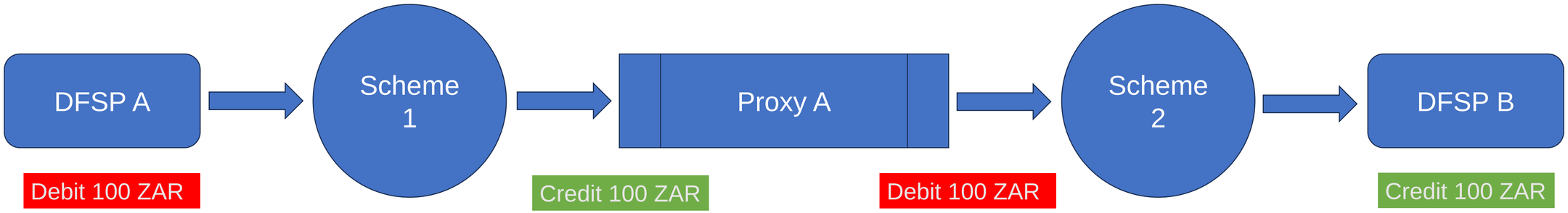
Balancing obligations

The requirement

- When a payment passes through a scheme, the scheme must record its passage.
- This is true even if none of the eventual parties to a payment are direct members of that scheme…
- … because we need to ensure that we can always trace the complete path of a payment through the system.
- The balances between all position accounts in any scheme must always net to zero.
  - This means: every credit in a scheme must be balanced by a corresponding debit.

Our proposal:

- Each proxy has position and settlement ledgers attached to it in both the Mojaloop schemes to which it belongs.
- For a given Mojaloop scheme, the title of the account is something like: “Funds due to or from the DFSPs represented by this proxy”.
- For any given transfer through a proxy, a debit in one of the schemes it is attached to will be exactly matched by a credit in the other scheme it is attached to.
- For instance, in a payment of 100 KES from DFSP A to DFSP B:



Managing time-outs

Principles of time-out management

- We time transfers out because we don’t want poor people’s funds to be unreachable when transactions may not complete.
- When a debtor DFSP requests execution of a payment, it specifies when the transfer should time out.
- Our principles:
  - There should be a single source of truth for time-out management. In a Mojaloop world, that’s the switch.
  - No participant should time a transfer out unilaterally.
  - When a participant asks after the status of a transfer, the answer may be “pending”.
  - This means: not yet finalised. A participant MUST wait until they can obtain a finalised state before timing a transfer out.
- How do we replicate these principles in an ecosystem where a transfer may pass through multiple switches?

Our proposal

- A switch should only enforce a time-out if it is the switch to which the creditor DFSP is directly attached.
- Requests for information about a transfer will be addressed to the creditor party in any case, so they will always be forwarded to that switch.
- Now we have a single source of truth again.

Liquidity cover

Liquidity cover

- It assures the creditor DFSP that it's safe to disburse its funds to the beneficiary immediately, because the debtor DFSP's funds have been reserved by the switch against known good funds.
- A requirement for a proxy is that it should be backed by good funds, administered by the schemes that it belongs to and available to both schemes.
  - These may be held in a single account, or in different accounts
  - Holding them in a single account is not a credit risk, since by definition a credit in one scheme is mirrored by a debit in another scheme and vice versa.
  - The good funds required by a proxy are in effect working capital. They are only required to allow schemes to settle independently of each other.
- But what if the debtor DFSP's good funds are known to some other scheme?

Proxies and accounts

- In each scheme, a proxy represents the net of all the transactions between that scheme and the other scheme(s) which the proxy represents.
- Its good funds account represents the net of all transactions between that scheme and the scheme(s) the proxy represents.
- It is administered, in effect, by the scheme to improve the efficiency of a multi-scheme system.
- So the creditor DFSP will rely on the liquidity check carried out by its scheme against its local proxy to warrant that sufficient funds are available to reimburse it for the payment it has executed.

An alternative plan

If schemes trust each other, then:

- Liquidity cover only needs to be held once, by the debtor DFSP.
- The rule for the switch is: only perform the liquidity check if the debtor DFSP is part of your scheme.
- A proxy’s good funds must be sufficient to settle in either scheme…
- … but there will never be a problem if both schemes settle.
- Position accounts for the intermediate proxies are allowed to exceed settled funds.

Reporting

Reporting

- Any participant should be able to see the full history of any transfer they were directly involved in.
  - For FXPs, this means the currency conversion only.
- Any scheme should be able to see the full history of any transfer that passed through their scheme.
- Every scheme and every participant will see the same message, since we’re not doing content conversion.
- Linked schemes need to have a merged reporting database, with proper segmentation to allow administrators to see only the parts they should have access to.

Any questions?