Stories to support inter-scheme Mojaloop

# Proxy registration

As a scheme operator, I want to know which proxies are attached to my system, so that I can broadcast queries on the multi-scheme ecosystem.

Extend the /services structures to allow a new type (PROXY), and associate the correct DFSPs with that type. For MVP, this can be done in configuration.

Ensure that a DFSP which has the type PROXY cannot have another type associated with it.

# Locality check

As a scheme operator, I want to know whether a given DFSP is:

1. A DFSP attached to my switch.
2. A Proxy attached to my switch.
3. Unknown to my switch.

So that I can take appropriate action when routing messages.

DFSPs can be identified either by ID or by IP address.

The list of proxies can be returned from 1 above.

# Time-out check

As a scheme operator in a multi-scheme ecosystem, I want to ensure that there is a single source of truth in relation to timing transfers out, so that I can avoid disputes arising from two participants in the system having a different view of the finalised status of a transaction.

* When a switch receives a transfer execution request, it should implement time-out checking until it has forwarded the request to a proxy and received a 202 response from the proxy. On receipt of the 202 response, it should turn time-out checking off.
* When a proxy receives a transfer execution request, it should implement time-out checking until it has forwarded the request to a switch and received a 202 response from the switch. On receipt of the 202 response, it should turn time-out checking off.

# Identifying the ID of a DFSP

We require a technique for reliably and securely identifying the proximate sender of a message, and resolving it to an FSP ID. A suggestion might be:

* Add an IP address to the set of information registered with a DFSP (or, perhaps a DFSP which is also a proxy).
* When each proxy is registered, ensure that its IP address is recorded.
* When a proxy receives a message, it adds its IP address to the *X-Forwarded-For* field in the header.
* When the switch receives a message, it interrogates each element in this field and checks the IP address against the list of recorded IP addresses.
* If the IP address belongs to a proxy:
  + Return the FSP ID associated with the IP address.
  + Exit the check.

The security risks associated with this approach (the IP address is passed in a field which does not (and cannot) form part of the signed header for non-repudiation purposes) are not considered to be problematic because:

* If another IP address is substituted for the one inserted by the proxy, it will not be found in the list of registered DFSPs.
* The content of the message continues to be protected by the original non-repudiation signature.

# Associating DFSPs with proxies

As a scheme operator, I want to be able to record an association between a proxy and a DFSP when I receive a message from that DFSP via a proxy, so that I can later route messages to that DFSP through the correct proxy.

The DFSP to be identified as the sender of the message is not the DFSP whose ID appears in the FSPIOP-Source parameter in the header. That parameter identifies the originator of the message. The DFSP to be identified is the proximate sender: the DFSP who directly forwarded the message to the switch, which may be a proxy. If the DFSP resolves to a proxy, then store the association between the FSP ID of the proxy and the FSP ID of the source DFSP as given in the FSPIOP-Source parameter in the header.

# Obligation creation

As a scheme operator, I want to ensure that the obligations for payments passing through my system are properly recorded, so that I can ensure that my books are balanced and my participants meet their obligations without the need to have recourse to disputes.

When recording obligations in a scheme:

* Replace DFSPs which do not belong to this scheme (per Section 2 above) with the proxies which represent them.
* If any obligations have the same proxy on both sides, do not record the obligation.

Note: this test is different from, and better than, the test originally proposed in the DA presentation. That test was correct for hub-and-spoke architectures such as COMESA, but would not have worked correctly for linear architectures such as those shown in topologies 2 and 3.

# DFSP resolution

As a scheme operator, I want to be able to identify the correct local route for a message intended for a given DFSP or DFSPs, so that I can route off-scheme messages to the correct destination.

This story needs to address two use cases:

* The ID of the destination DFSP is known, but it is not a member of the local scheme (as tested by: does its FSP ID appear in the list of local participants?) and there is no proxy association for it (see Section 5 above).
* The ID of the destination DFSP is not known, but it is known not to be a DFSP which is a member of the local scheme (e.g. it is an address resolution call and the appropriate oracle has returned a NOT FOUND).

In both use cases, the following actions should be performed:

When a message is received from a DFSP:

1. If the URI/FSP ID combination is not already the subject of a search, store:
   1. The URI of the message.
   2. The FSP ID of the eventual recipient of the message, if it is known.
2. Obtain a list of all the proxies attached to the scheme, by interrogating the list maintained per Section 1 above.
3. Forward a copy of the message received to all proxies in the list.
   1. If the message was received from a proxy, do not send the message to that proxy.
4. If the number of messages sent was non-zero, store the number of messages sent against the URI/FSP ID entry.
5. If the number of messages sent was zero, return an error 3201: *Destination FSP Error*.

When a message is received from a proxy (see Section 5 above):

* Search the list of searches for:
  + Either a search with the DFSP and URI returned via the proxy.
  + Or a search with a blank DFSP and the URI returned via the proxy.
* If neither is found, end: we’re not doing this search.
* Decrement the counter created in step 4 above.
* If the counter is now zero, remove the search from the list of searches.
* If the message returned is a 3201 error (DFSP not found):
  + If the counter is now zero, forward the message to the next destination.
* Otherwise (DFSP was found):
  + Forward the message to the next destination.
* End.

# Services extension

As a DFSP, I want to know which participants offer FXP services anywhere on a network of connected schemes, so that I can obtain currency conversion from the widest possible range of candidates.

When a **GET /services/FXP/aaa/bbb** call is issued, the switch should initiate the DFSP search across the whole interconnected network. This process is analogous to the DFSP resolution process described in Section 7 above; but there are sufficient differences between the two processes to make a separate description of the service identification process useful.

To identify the providers of a service, the following actions should be performed:

1. Obtain a list of the providers of the service local to the switch.
2. Obtain a list of all the proxies attached to the scheme, by interrogating the list maintained per Section 1 above.
3. Forward a copy of the message received to all proxies in the list.
   1. If the message was received from a proxy, do not send the message to that proxy.
4. If the number of messages sent was non-zero, store the number of messages sent against the URI/FSP ID entry.
5. Store the list of local service providers against the URI/FSP Id entry.
6. If the number of messages sent was zero, return a PUT response to the request with the list of local service providers stored in step 5 above.

When a message is received from a proxy (see Section 5 above), the following actions should be performed:

* Search the list of searches for a search with the DFSP and URI returned via the proxy.
* Decrement the counter created in step 4 above.
* If the counter is now zero:
  + Add the DFSP content stored against the query to the DFSP content of the returned message.
  + Forward the response to the requester.
  + Remove the search from the list of searches.
  + End.
* Otherwise:
  + Add the DFSPs returned in the message to the list of DFSPs stored against the search.
  + End.

# Liquidity check

As a scheme administrator, I want to be able to decide whether to trust upstream schemes to perform the liquidity check on my behalf if the debtor party to a statement is not a member of my scheme, so that I can optimise the use of working capital in my system.

On receipt of a **POST /transfers** message, if the debtor party is not a member of my scheme, only perform the liquidity check if a configuration flag (as it might be, Allow\_Upstream\_Liquidity\_Checking) is set to FALSE.