The Problems

1. As currently implemented, the Mojaloop Bulk Payment service operates in a manner that doesn't match market expectations
   - Payers just want to hand over a list; Mojaloop wants a series of sorted lists, one per payee DFSP

2. Payee DFSPs vary in their capabilities; some can accept a bulk payments list, others cannot

Short Term Solution

Accept a bulk payments list from a DFSP, and deal with them as though they're individual transactions:

- *Process them one by one (discover, agree, transfer)*

- *Assemble a single bulk transfer response and return it to the sender*

Pros:
- Simple, straightforward to implement
- Supports some DFSPs' inability to accept bulk lists

Cons:
- Increased load on the Hub

- Only suitable for relatively short lists; good for loan disbursements, not good for ECT

Medium Term Solution

Accept a bulk payments list from a DFSP, and sort it into individual lists (one per payee DFSP):

- *Process each list as a Mojaloop Bulk Payment*
  - *For some Payee DFSPs, it will be necessary to break up the list at the "receive" end and process as individual transactions*
- *Assemble a single bulk transfer response and return it to the sender*

Pros:

- Allows us to reach scale
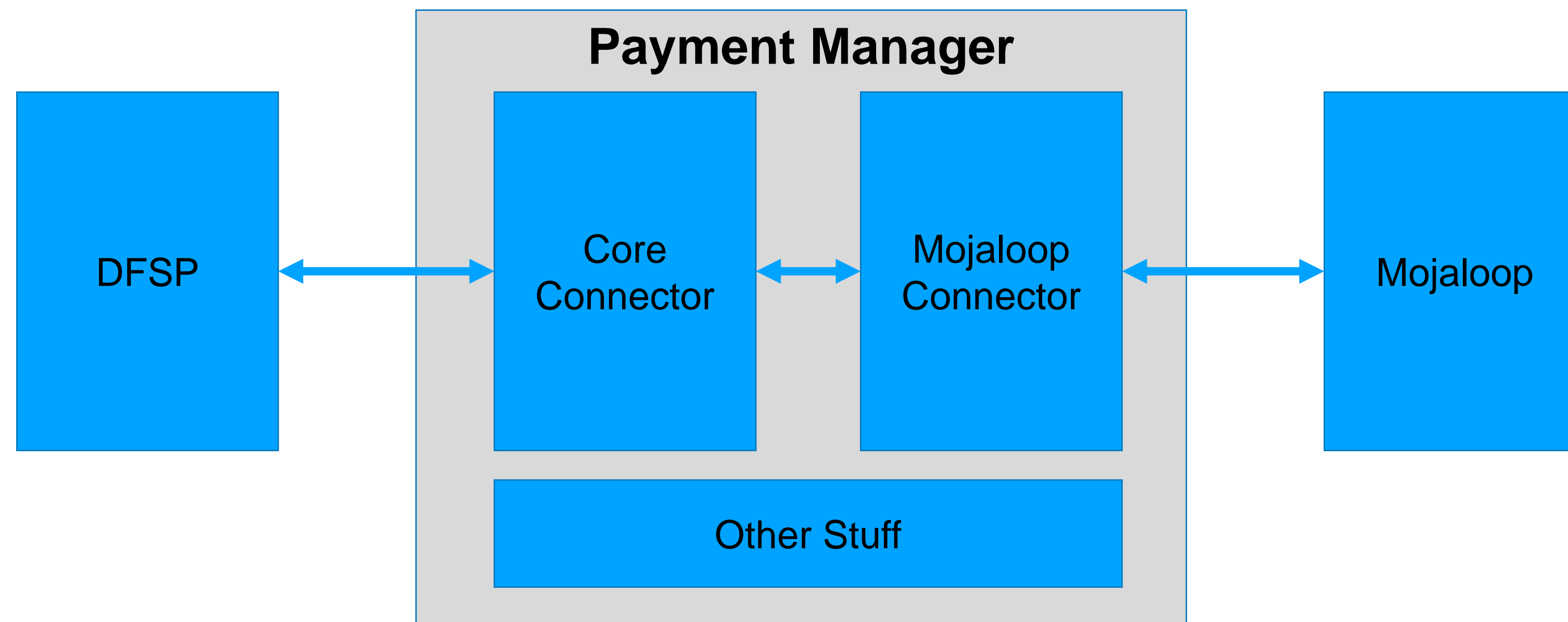- Supports some DFSPs' inability to accept bulk lists

Cons:

- More complex to implement; longer lead time

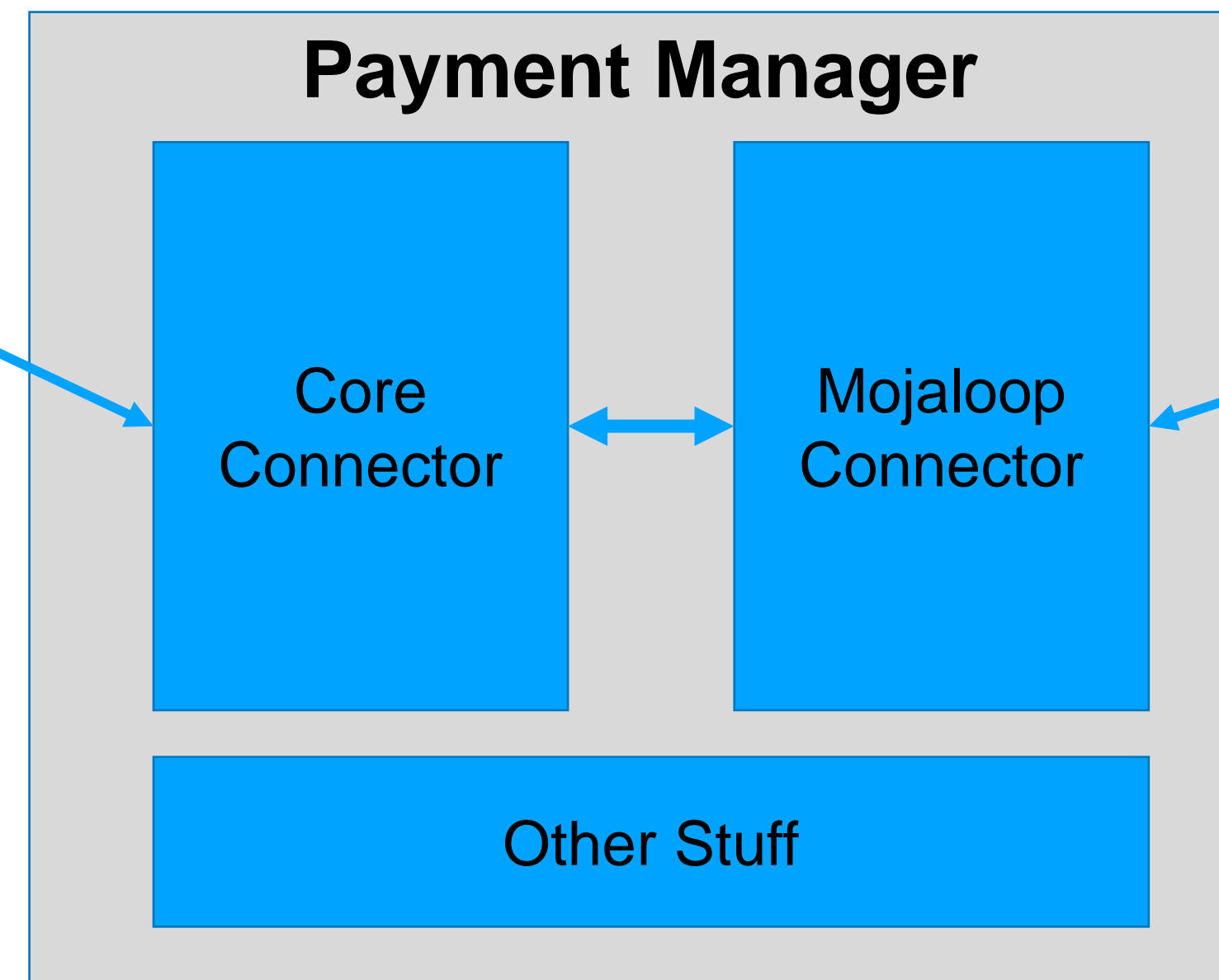# Route to Implementation of the Short Term Solution

- This is a core use case for WynePay, so the STS has been fast tracked
- WynePay uses Payment Manager to connect DFSPs, which gives us a route to implementation: using Payment Manager for orchestration of bulk payments

Payment Manager

DFSP ⟷ Core Connector ⟷ Mojaloop Connector ⟷ Mojaloop

Other Stuff

Route to Implementation of the Short Term Solution

Route to Implementation of the Short Term Solution

Payment Manager

Specific to WynePay

OSS Implementation

Core Connector

Mojaloop Connector

Routing:
"Wallet shortcode, MSISDN"

Other Stuff
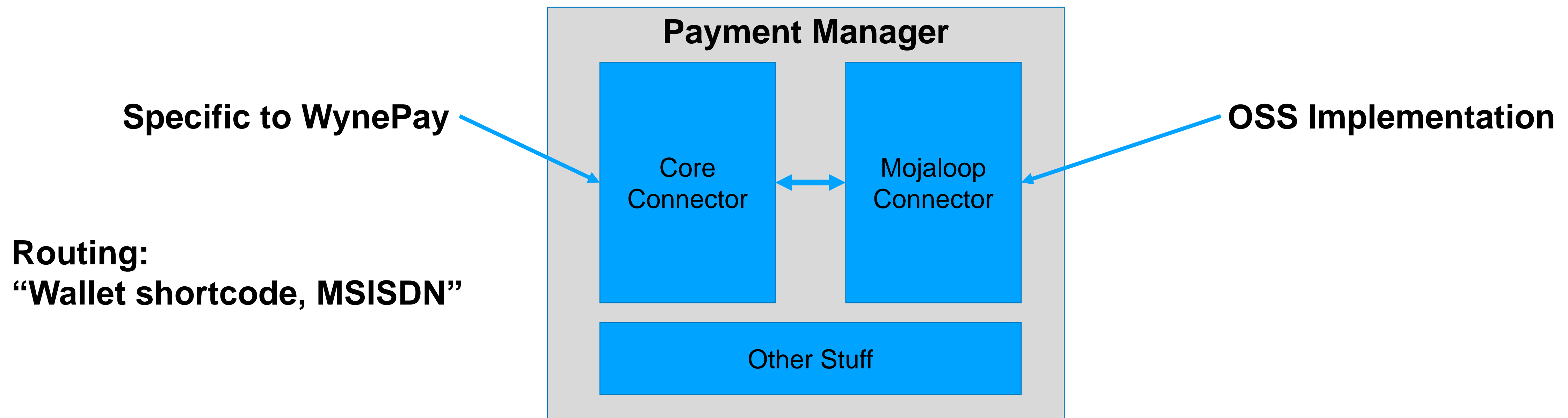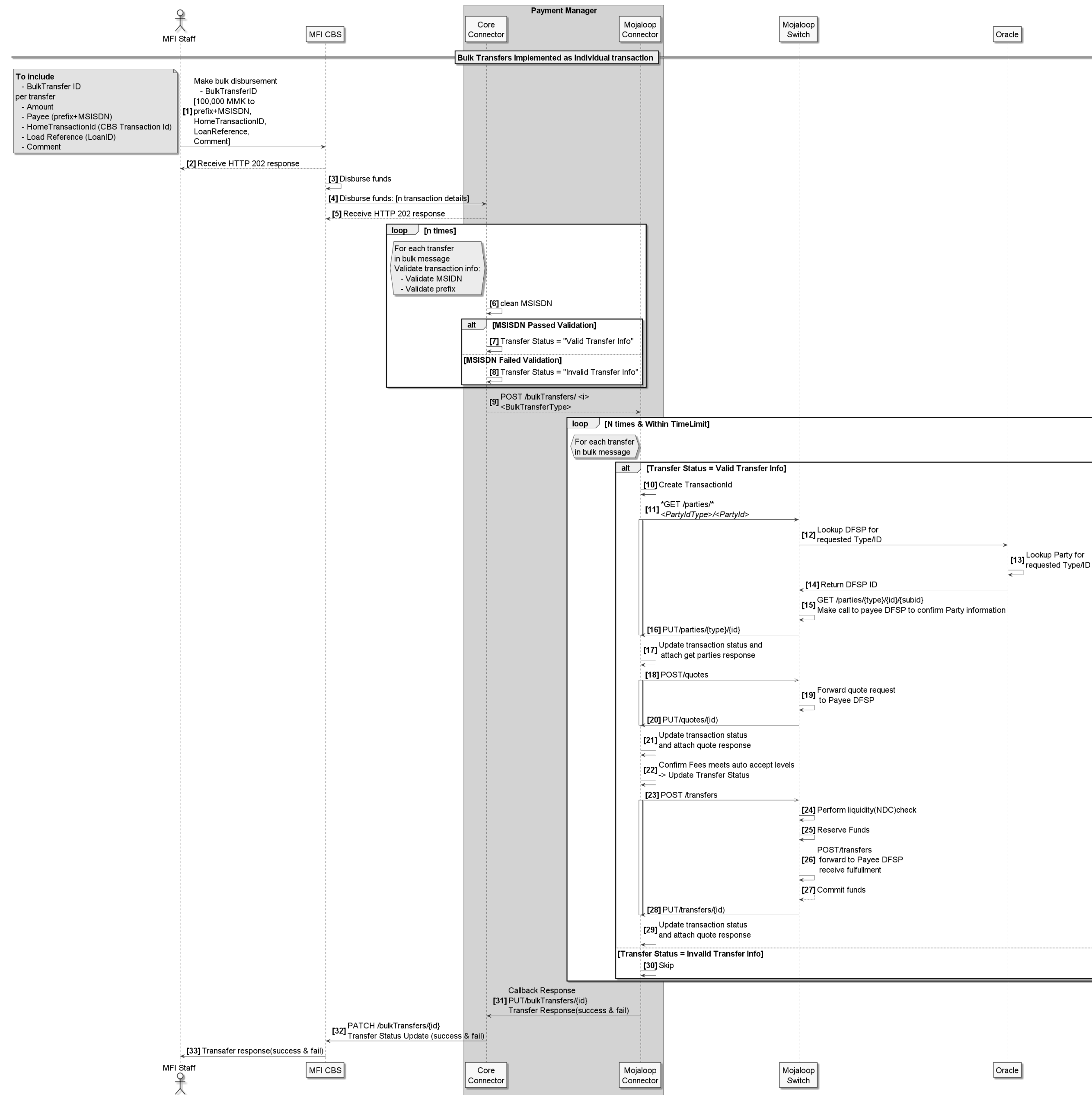
Process Flow
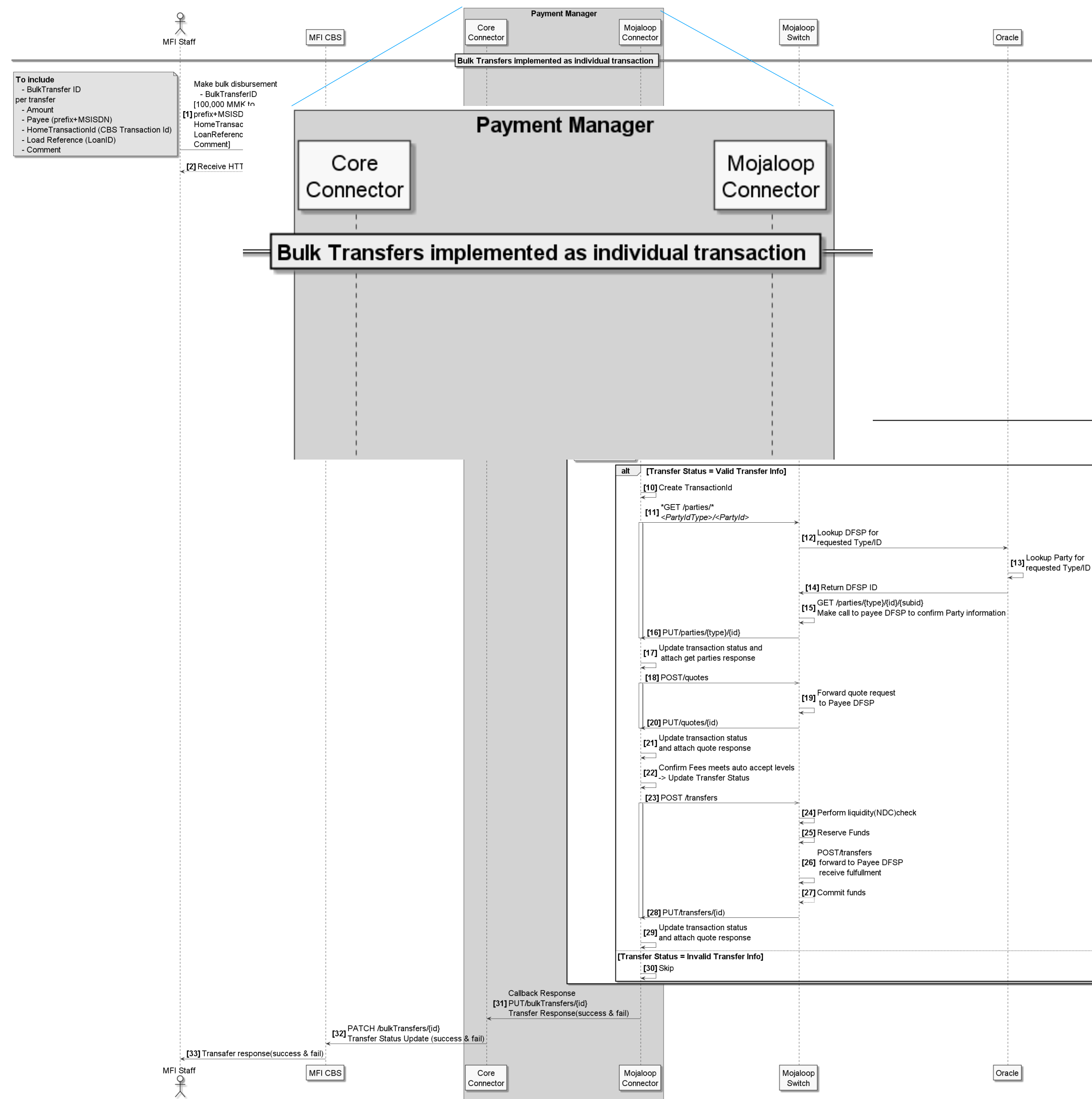
Process Flow

Process Flow: Core Connector
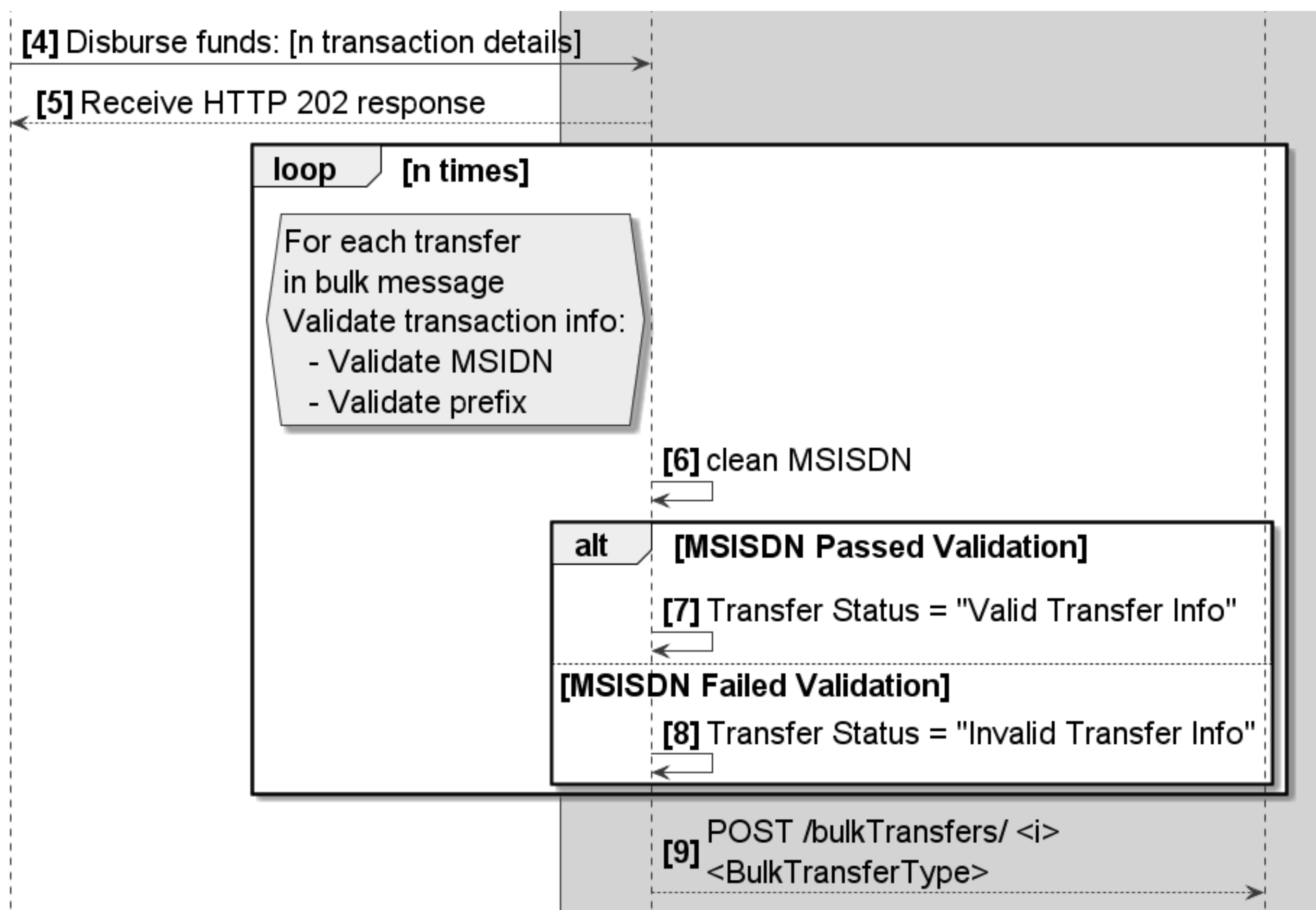
Process Flow: Mojaloop Connector

loop   [N times & Within TimeLimit]

For each transfer in bulk message

alt   [Transfer Status = Valid Transfer Info]

[10] Create TransactionId

[11] *GET /parties/* <PartyIdType>/<PartyId>

[12] Lookup DFSP for requested Type/ID

[13] Lookup Party for requested Type/ID

[14] Return DFSP ID

[15] GET /parties/{type}/{id}/{subid} Make call to payee DFSP to confirm Party information

[16] PUT/parties/{type}/{id}

[17] Update transaction status and attach get parties response

[18] POST/quotes

[19] Forward quote request to Payee DFSP

[20] PUT/quotes/{id}

[21] Update transaction status and attach quote response

[22] Confirm Fees meets auto accept levels -> Update Transfer Status

[23] POST /transfers

[24] Perform liquidity(NDC)check

[25] Reserve Funds

[26] POST/transfers forward to Payee DFSP receive fulfullment

[27] Commit funds

[28] PUT/transfers/{id}

[29] Update transaction status and attach quote response

[Transfer Status = Invalid Transfer Info]

[30] Skip

11

Progress

We have a design; it's been reviewed by the dev teams, and all of the user stories have been documented

- Sized, acceptance criteria defined, tests defined, etc.

Implementation has begun, and will be completed this PI

Progress

We have a design; it's been reviewed by the dev teams, and all of the user stories have been documented

• Sized, acceptance criteria defined, tests defined, etc.

Implementation has begun, and will be completed this PI

Unanticipated Bonus:

During the design process, it became
clear that the most straightforward route
to implementation included the upgrade
of the oracle to support P2P transactions

Thank you!

Comments/Questions?