Inclusive Instant Payment System (IIPS) Integration

Introduction

When integrating legacy core banking systems into a modern, real-time, push based transaction flow there can be some challenges. These are largely based on what vendor provided integration API's are on offer. This document discusses what an ideal integration should look like, some of the typical limitations seen in vendor API's, and flow patterns that are used to overcome some of these limitations. The mitigation mechanisms and implications of each approach are discussed.

Payer DFSP Integration vs Payee DFSP Integration

The Payer DFSP organization and the Payee DFSP organization in a transfer need to provide very different integrations, and take on different responsibilities in a transfer. As a result this document will focus on integrations as first a Payer organization, and then as a Payee organization. It is also important to note that all integrations built need to support both payer and payee integrations even if the use case only dictates a single direction. This is because inside Mojaloop the only mechanism for reversing a transfer is to initiate a transfer in the reverse direction. Mojaloop supports push payments which aligns with the level-one principle aimed at achieving financial inclusion.

Two phase (Reserve and Commit) transfer vs Single phase committed transfer

To provide some context, the anatomy of a Mojaloop transfer the two phase (reserve and Commit) transfer are integration steps that occur in the Mojaloop transfer phase. Making use of the two step reserve & commit transfer is the ideal for building integrations. This is because both systems can perform reservations before funds are committed. This pattern maximizes use of liquidity, minimizes the time funds are reserved from the payer and thereby reduces reconciliation effort. Often the core-banking system in use does not provide this functionality, meaning an alternative approach is needed. This document starts by first describing the integration patterns that use the ideal reserve-commit integration, and then discusses approaches that can be taken for integrating single phase commit transfers. Some of these are anti-patterns and should not be used. These are discussed too.

Define use-case based API’s

Because Scheme rules and regulations require certain payment use-cases to have additional fields that are easier to describe and enforce through API definitions, it is recommended to standardize the Payer DFSP’s API that initiates the payment for different use-cases. This approach can be extended to the payee DFSP’s API if found necessary based on the requirements.

Payer DFSP Integration Patterns

Reserve-Commit vs Transfer and Reversal Integrations

Both integration patterns discussed below show the reserve-commit integration as the Payer DFSP organization the reserve. If a reserve-commit is not available, then the following integrations can be used as a substitute.

1. Reservation in Payer's Account → Transfer from Payer’s Account
2. Finalize transfer into Payer’s Account → No action necessary
3. Rollback transfer into Payers Account → Reverse the transfer in step 1. which should return funds to Payer’s Account

Number of Integration points with Core Banking system

The payer DFSP integration can choose how many points of interaction are required with the core banking system. These points of integration align with the three phases of a Mojaloop transfer, and allow the core-banking system to make checks and confirmations before proceeding with the final transfer phase. These can be described as integrations to:

1. Initiate the transfer
2. Confirm the party
3. Confirm the terms and conditions of the cross-currency conversion
4. Confirm the terms and conditions of the transfer
5. Notify the payer of the transfer state I.e. is it complete or aborted

Most core-banking systems don't support integrations at all of these types of integrations, so this document will discuss two integration patterns. The first pattern is called Payer DFSP Confirmation Integration where steps 2, 3 and 4 are combined into a single confirmation of all terms before proceeding. The second pattern is a Payer DFSP Single Integration where no confirmation of the transfer terms are integrated into the core-banking system.

Payer DFSP Confirmation Integration Pattern

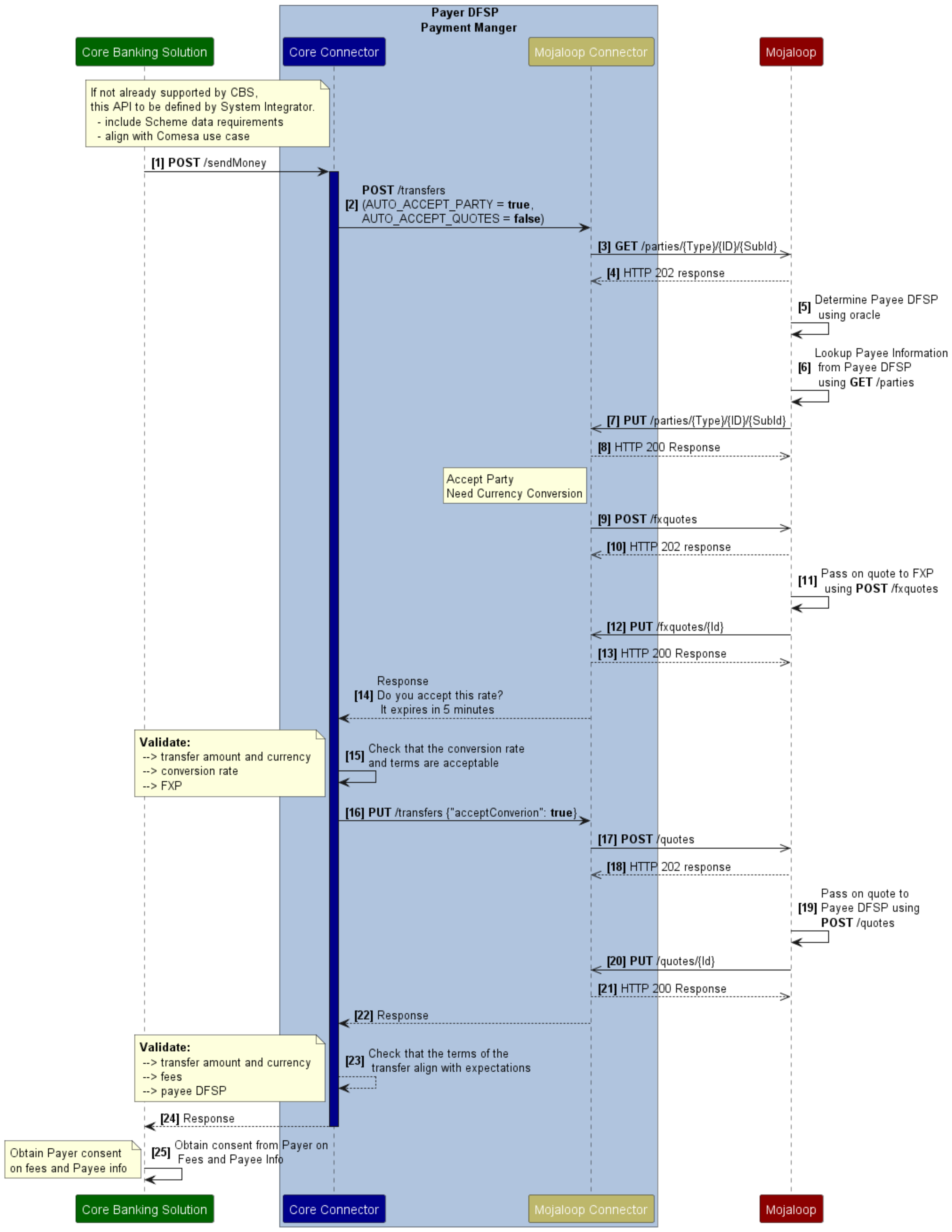
This pattern consists of two parts namely obtaining the consent part and executing the transfer part. The first obtaining consent part takes the initiated payment, and returns all the terms of the transfer so that they can be confirmed by the core banking system. I.e. it provides an integration to support the Payer being presented with the terms of the transfer before proceeding with the transfer. This is a Mojaloop and level one principle I.e. transparent fees. The transfer terms include the Payee information, the currency conversion information, and the associated fees. The transfer part of this pattern begins if the payer agrees to these terms.

Obtaining Consent

There are three important integration considerations when building this part of the integration. Firstly the initiating API design. It is recommended that this aligns with a scheme use case, and includes all required fields and expectations of the scheme. I.e. clear assignment of which field/parameters are optional and what type definition they hold. If this is an ISP20022 API, then it is recommended that expected fields/parameters are marked accordingly and a definition of the Market implementation of that API is still provided for that use-case

Secondly, currency conversion terms should be validated. The foreign exchange provider the amount and the currency and conversion rate should be checked to make sure that they align with expectations and are acceptable to proceed.

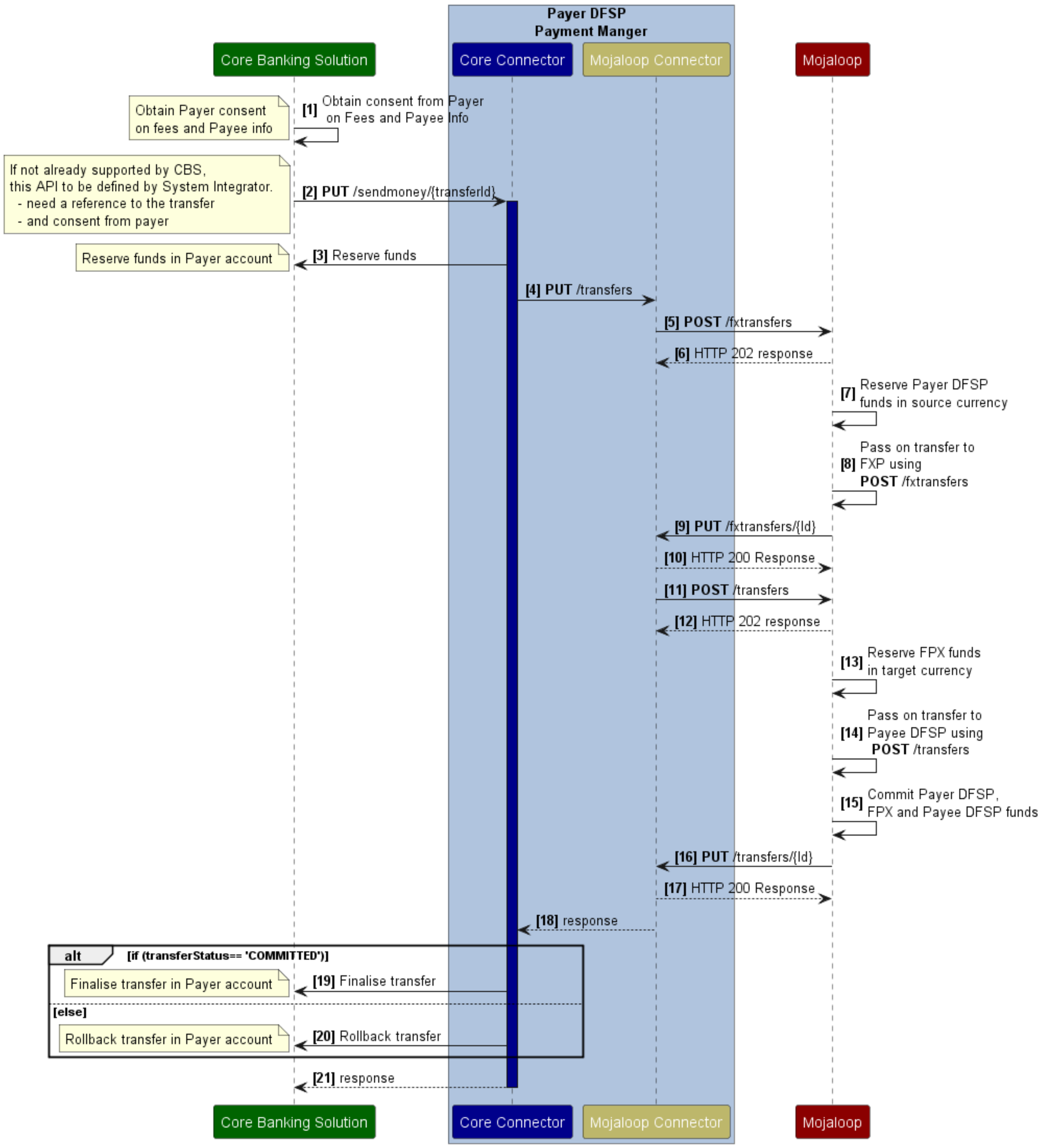
Thirdly, the transfer terms should be validated. The payee DFSP organization should be validated. I.e. check to ensure that this agreement is being made with the organization that you expect. The transfer amount should be checked. This is the amount that will be deducted from the pre-funded liquidity and ultimately transferred to the FXP and Payee organization. The fees charged should be checked. I.e. to ensure that they are reasonable and within expectations. The integration can now return this validated information back to the core banking system to ask for confirmation of the transfer terms.



Payer DFSP
Payment Manger
Core Banking Solution
Core Connector
Mojaloop Connector
Mojaloop
If not already supported by CBS, this API to be defined by System Integrator. - include Scheme data requirements - align with Comesa use case
[1] POST /sendMoney
[2] POST /transfers (AUTO_ACCEPT_PARTY = true, AUTO_ACCEPT_QUOTES = false)
[3] GET /parties/{Type}/{ID}/{SubId}
[4] HTTP 202 response
[5] Determine Payee DFSP using oracle
[6] Lookup Payee Information from Payee DFSP using GET /parties
[7] PUT /parties/{Type}/{ID}/{SubId}
[8] HTTP 200 Response
Accept Party
Need Currency Conversion
[9] POST /fxquotes
[10] HTTP 202 response
[11] Pass on quote to FXP using POST /fxquotes
[12] PUT /fxquotes/{Id}
[13] HTTP 200 Response
[14] Response Do you accept this rate? It expires in 5 minutes
Validate: --> transfer amount and currency --> conversion rate --> FXP
[15] Check that the conversion rate and terms are acceptable
[16] PUT /transfers {"acceptConverion": true}
[17] POST /quotes
[18] HTTP 202 response
[19] Pass on quote to Payee DFSP using POST /quotes
[20] PUT /quotes/{Id}
[21] HTTP 200 Response
[22] Response
Validate: --> transfer amount and currency --> fees --> payee DFSP
[23] Check that the terms of the transfer align with expectations
[24] Response
Obtain Payer consent on fees and Payee info
[25] Obtain consent from Payer on Fees and Payee Info

Executing the Transfer

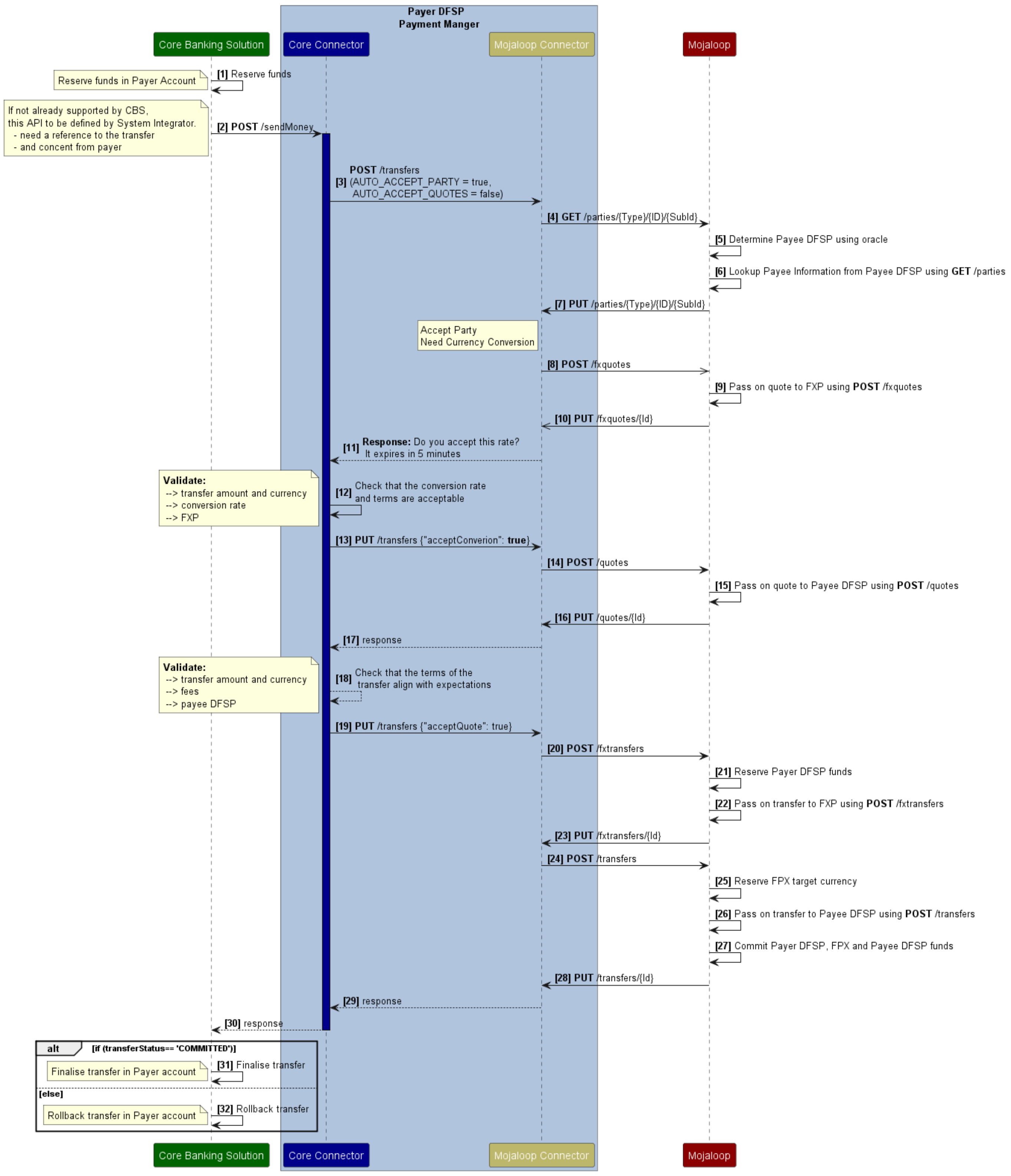
The executing the transfer part of this integration pattern begins when the Payer accepts the terms of the transfer. A reservation of the funds in the Payer's account must be made before transfer is initiated. When notification of the transfer state is received, then the transfer from the Payer's account should either be finalized or rolled back based on whether the transfer state is committed or not.


Payer DFSP
Payment Manger
Core Banking Solution
Core Connector
Mojaloop Connector
Mojaloop
Obtain Payer consent on fees and Payee info
[1] Obtain consent from Payer on Fees and Payee Info
If not already supported by CBS, this API to be defined by System Integrator. - need a reference to the transfer - and consent from payer
[2] PUT /sendmoney/{transferId}
Reserve funds in Payer account
[3] Reserve funds
[4] PUT /transfers
[5] POST /fxtransfers
[6] HTTP 202 response
[7] Reserve Payer DFSP funds in source currency
[8] Pass on transfer to FXP using POST /fxtransfers
[9] PUT /fxtransfers/{Id}
[10] HTTP 200 Response
[11] POST /transfers
[12] HTTP 202 response
[13] Reserve FPX funds in target currency
[14] Pass on transfer to Payee DFSP using POST /transfers
[15] Commit Payer DFSP, FPX and Payee DFSP funds
[16] PUT /transfers/{Id}
[17] HTTP 200 Response
[18] response
alt [if (transferStatus== 'COMMITTED')]
Finalise transfer in Payer account
[19] Finalise transfer
[else]
Rollback transfer in Payer account
[20] Rollback transfer
[21] response

Payer DFSP Single Integration

The single payer DFSP integration pattern is appropriate if the core-banking system is not able to ask for payers consent before proceeding with the transfer. This is not ideal when fees are involved, as this is a level-one principle that is not being addressed. So all attempts to not use this pattern, should be made. There are however some use cases where the use of this pattern is unavoidable. In those cases, the premise that the integration should take, is to then take on the responsibility for those checks in the integration. I.e. The checks on the payee details, terms of the conversion, terms of the transfer still need to be taken, however they are made based on an agreement with the payer to be within certain parameters that the integration design is configured to enforce. The question that need to be answered are:

1. What validation checks can be made to verify the payee details?
2. How is the conversion rate validated?
3. How do I judge whether the fees are appropriate, and what checks must be made to validate the transfer amount.


Payer DFSP
Payment Manger
Core Banking Solution
Core Connector
Mojaloop Connector
Mojaloop
[1] Reserve funds
Reserve funds in Payer Account
If not already supported by CBS, this API to be defined by System Integrator.
- need a reference to the transfer
- and concent from payer
[2] POST /sendMoney
[3] POST /transfers (AUTO_ACCEPT_PARTY = true, AUTO_ACCEPT_QUOTES = false)
[4] GET /parties/{Type}/{ID}/{SubId}
[5] Determine Payee DFSP using oracle
[6] Lookup Payee Information from Payee DFSP using GET /parties
[7] PUT /parties/{Type}/{ID}/{SubId}
Accept Party
Need Currency Conversion
[8] POST /fxquotes
[9] Pass on quote to FXP using POST /fxquotes
[10] PUT /fxquotes/{Id}
[11] Response: Do you accept this rate? It expires in 5 minutes
Validate:
--> transfer amount and currency
--> conversion rate
--> FXP
[12] Check that the conversion rate and terms are acceptable
[13] PUT /transfers {"acceptConverion": true}
[14] POST /quotes
[15] Pass on quote to Payee DFSP using POST /quotes
[16] PUT /quotes/{Id}
[17] response
Validate:
--> transfer amount and currency
--> fees
--> payee DFSP
[18] Check that the terms of the transfer align with expectations
[19] PUT /transfers {"acceptQuote": true}
[20] POST /fxtransfers
[21] Reserve Payer DFSP funds
[22] Pass on transfer to FXP using POST /fxtransfers
[23] PUT /fxtransfers/{Id}
[24] POST /transfers
[25] Reserve FPX target currency
[26] Pass on transfer to Payee DFSP using POST /transfers
[27] Commit Payer DFSP, FPX and Payee DFSP funds
[28] PUT /transfers/{Id}
[29] response
[30] response
alt
[if (transferStatus== 'COMMITTED')]
Finalise transfer in Payer account
[31] Finalise transfer
[else]
Rollback transfer in Payer account
[32] Rollback transfer
Core Banking Solution
Core Connector
Mojaloop Connector
Mojaloop

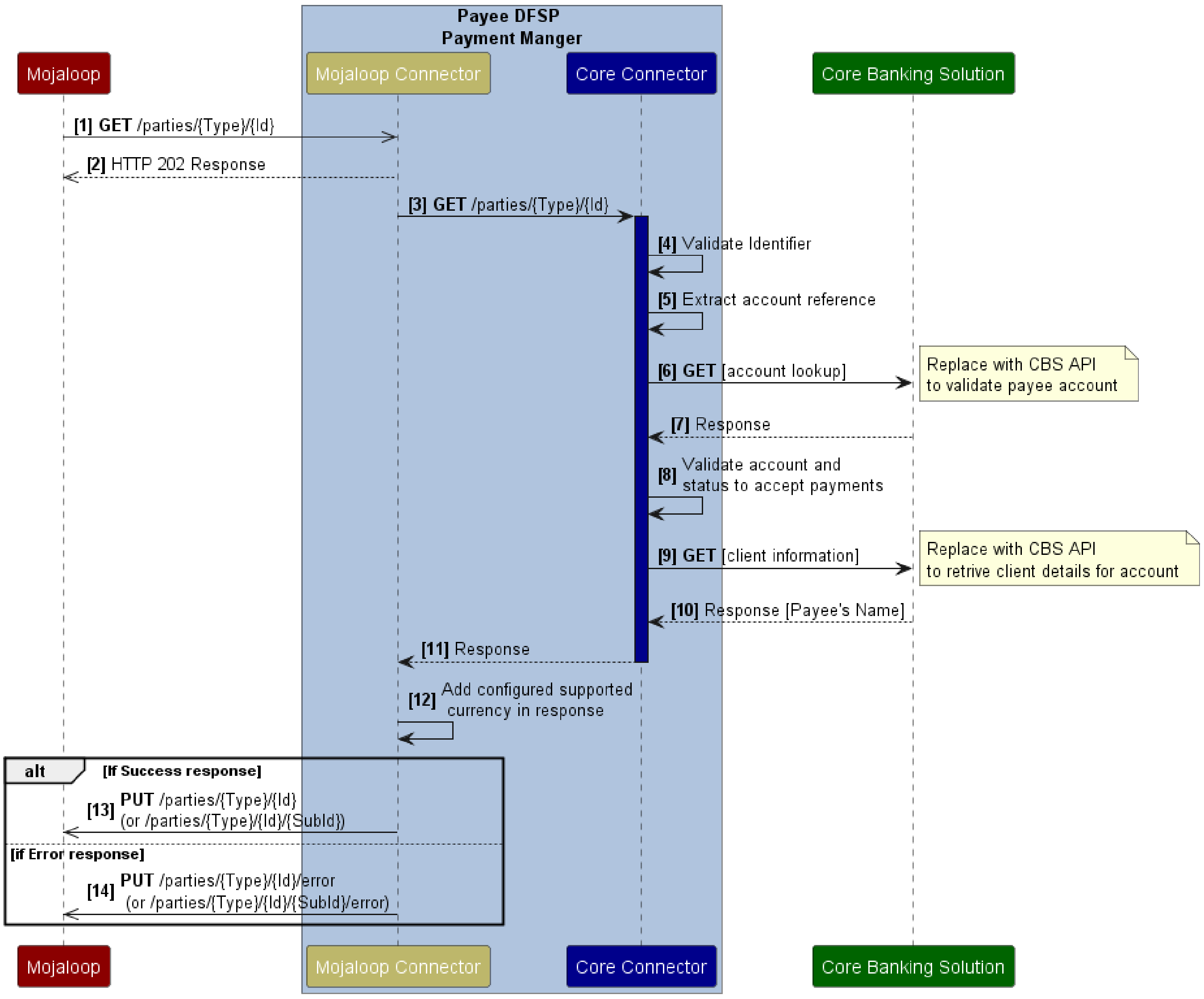
Payee DFSP Integration Patterns

Three points of Integration points with Core Banking system The payee DFSP integration must respond to each phase of a mojaloop transfer, so therefore has to do a three phase integration with the core-banking system. There are certain functions that a Payee DFSP must assume in each of the phases. Let’s discuss those now.

1. **Discovery**
   In Discovery the payee DFSP is presented with a payee payment identifier. The payee DFSP is required to validate that this payee identifier does refer to an account controlled by the payee DFSP, and that this account is active and can accept funds. The payee DFSP additionally specifies the currency/s in which this account can accept funds in. If these requirements are met, the payee DFSP is then obligated to return some identifiable information that the Payer can use to validate that this is the correct account. E.g. the account holder’s name.
2. **Agreement**
   In Agreement the payee DFSP must perform all validations required to perform the transfer. For example AML check, KYC checks and transfer threshold limit checks. By assenting to the terms of a payment, the payee DFSP commits to crediting the funds to the beneficiary, provided the instruction is received before the expiry time and the payee DFSP is not prevented by force majeure (e.g. if the beneficiary's account has been suspended in the interim.) The payee DFSP then compiles the terms of the transfer which includes the fees. These terms become the bases upon which the transfer is executed.
3. **Transfer**
   If the payer DFSP’s customer agrees to the terms of the transfer, then the transfer phase is started. The payee DFSP is most at risk during this phase, as it is the payee DFSP that releases funds to the Payee. The payee DFSP is required to provide the cryptographic fulfillment of the terms to the hub before the transfer timeout. If these conditions are met, then the hub will allocate the pre-funded liquidity to cover that transfer which gets distributed into the liquidity accounts during the settlement process.

Payee DFSP Discovery Integration Pattern

There is a single pattern for building payee DFSP Discovery integrations.



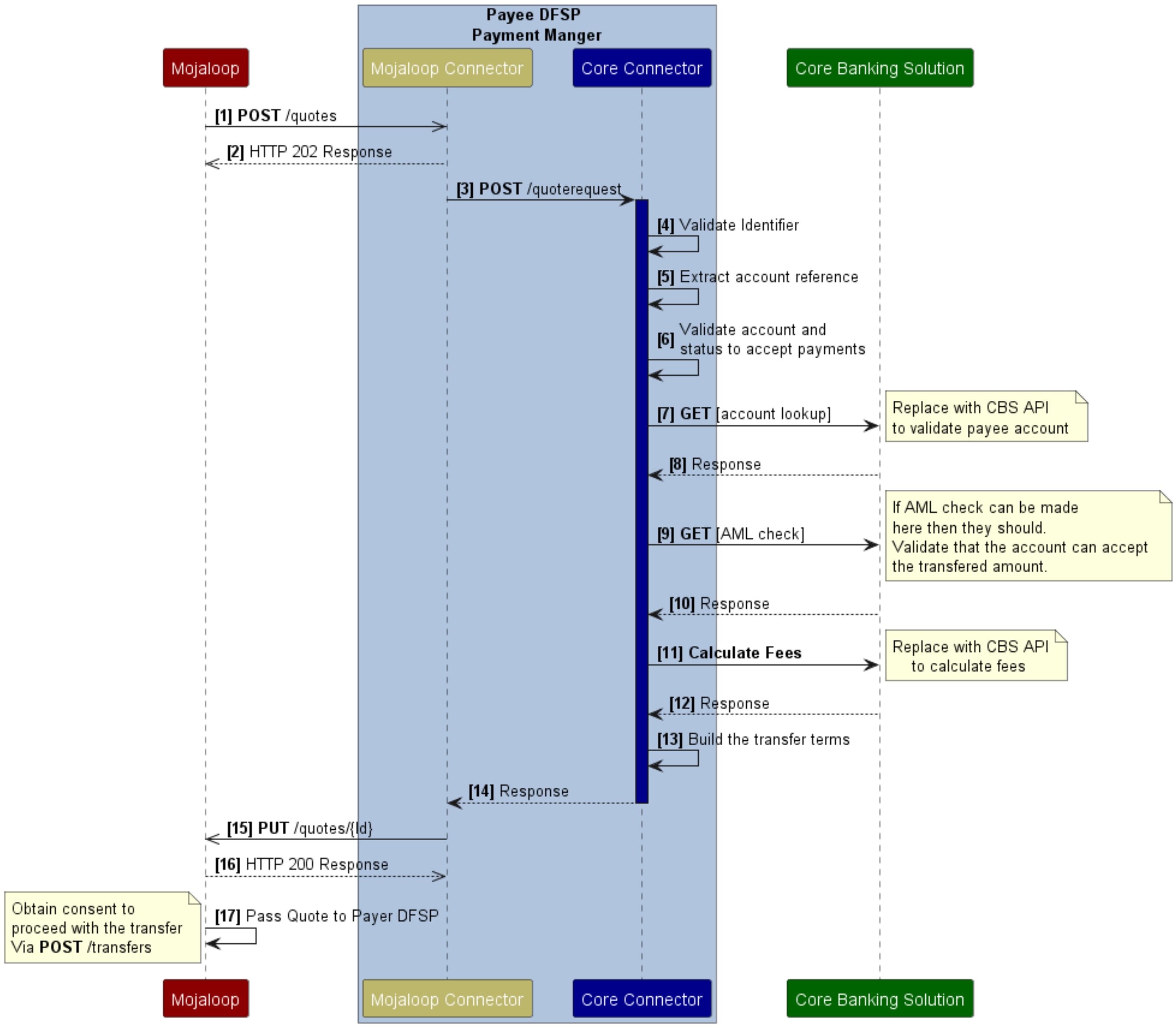
Payee DFSP
Payment Manger
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution
[1] GET /parties/{Type}/{Id}
[2] HTTP 202 Response
[3] GET /parties/{Type}/{Id}
[4] Validate Identifier
[5] Extract account reference
[6] GET [account lookup]
Replace with CBS API to validate payee account
[7] Response
[8] Validate account and status to accept payments
[9] GET [client information]
Replace with CBS API to retrive client details for account
[10] Response [Payee's Name]
[11] Response
[12] Add configured supported currency in response
alt
[If Success response]
[13] PUT /parties/{Type}/{Id} (or /parties/{Type}/{Id}/{SubId})
[if Error response]
[14] PUT /parties/{Type}/{Id}/error (or /parties/{Type}/{Id}/{SubId}/error)

The function that need to be catered for are:

1. Validating Supported Scheme Payment Identifiers
2. Extracting Account Information from the Payment Identifiers
3. Validate the account by checking that it exists, is active and can receive funds in the configured currency
4. Query the client details for that account

Payee DFSP Agreement Integration Pattern

There is a single pattern for building payee DFSP Agreement integrations. Here it is.


Payee DFSP
Payment Manger
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution
[1] POST /quotes
[2] HTTP 202 Response
[3] POST /quoterequest
[4] Validate Identifier
[5] Extract account reference
[6] Validate account and status to accept payments
[7] GET [account lookup]
Replace with CBS API to validate payee account
[8] Response
[9] GET [AML check]
If AML check can be made here then they should. Validate that the account can accept the transfered amount.
[10] Response
[11] Calculate Fees
Replace with CBS API to calculate fees
[12] Response
[13] Build the transfer terms
[14] Response
[15] PUT /quotes/{Id}
[16] HTTP 200 Response
[17] Pass Quote to Payer DFSP
Obtain consent to proceed with the transfer Via POST /transfers

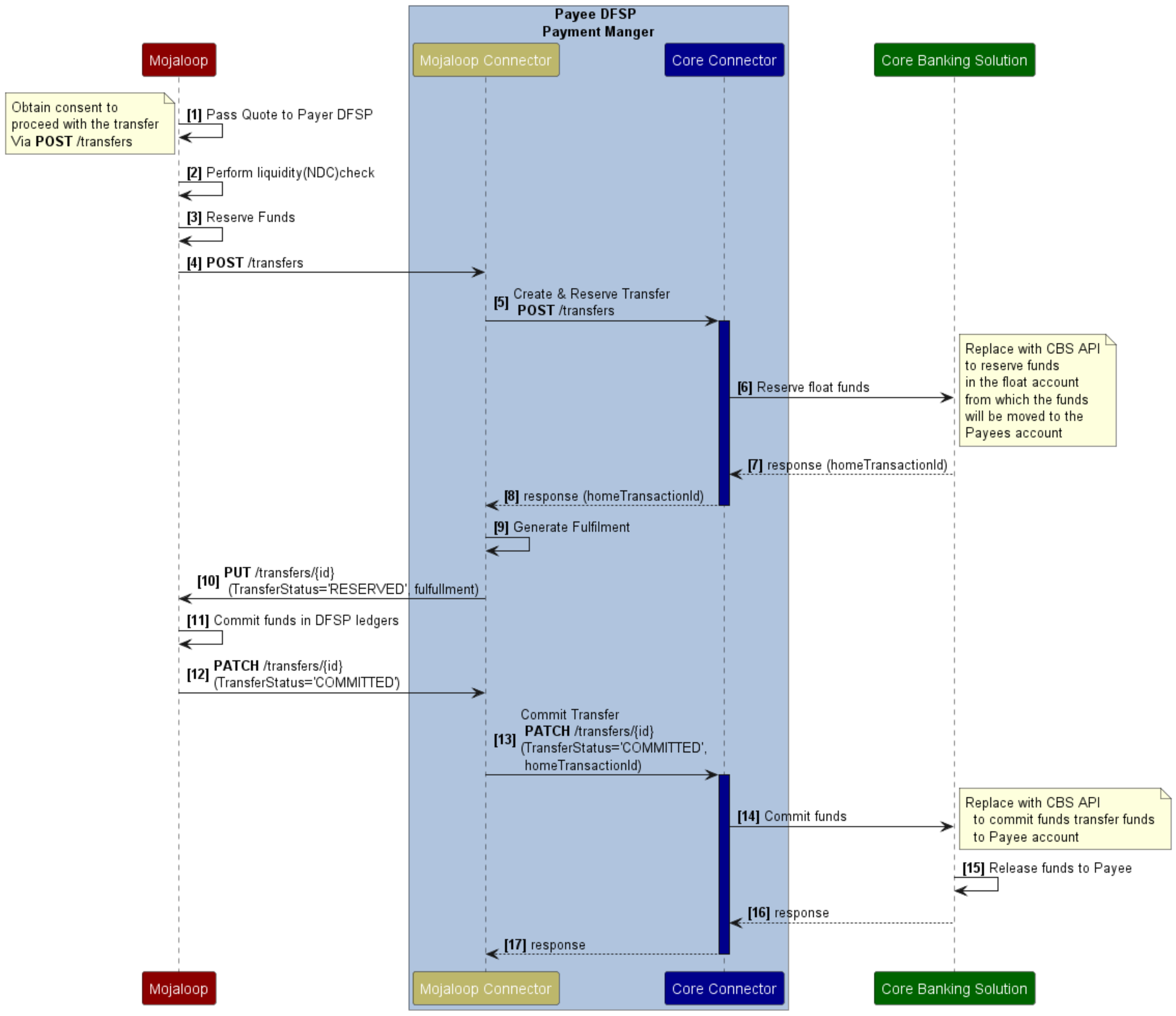
The Payer DFSP is required to:

1. Validate the Payer DFSP organization.
2. Validate the payment payee identifier.
3. Validate that the account is in good standing and can receive funds in the provided currency.
4. Perform all compliance checks which may include AML.
5. Calculate any fees that must be included in the terms of the transfer.
6. Build the terms of the transfer.

The terms of the transfer are then returned.

Payee DFSP Transfer Integration Pattern

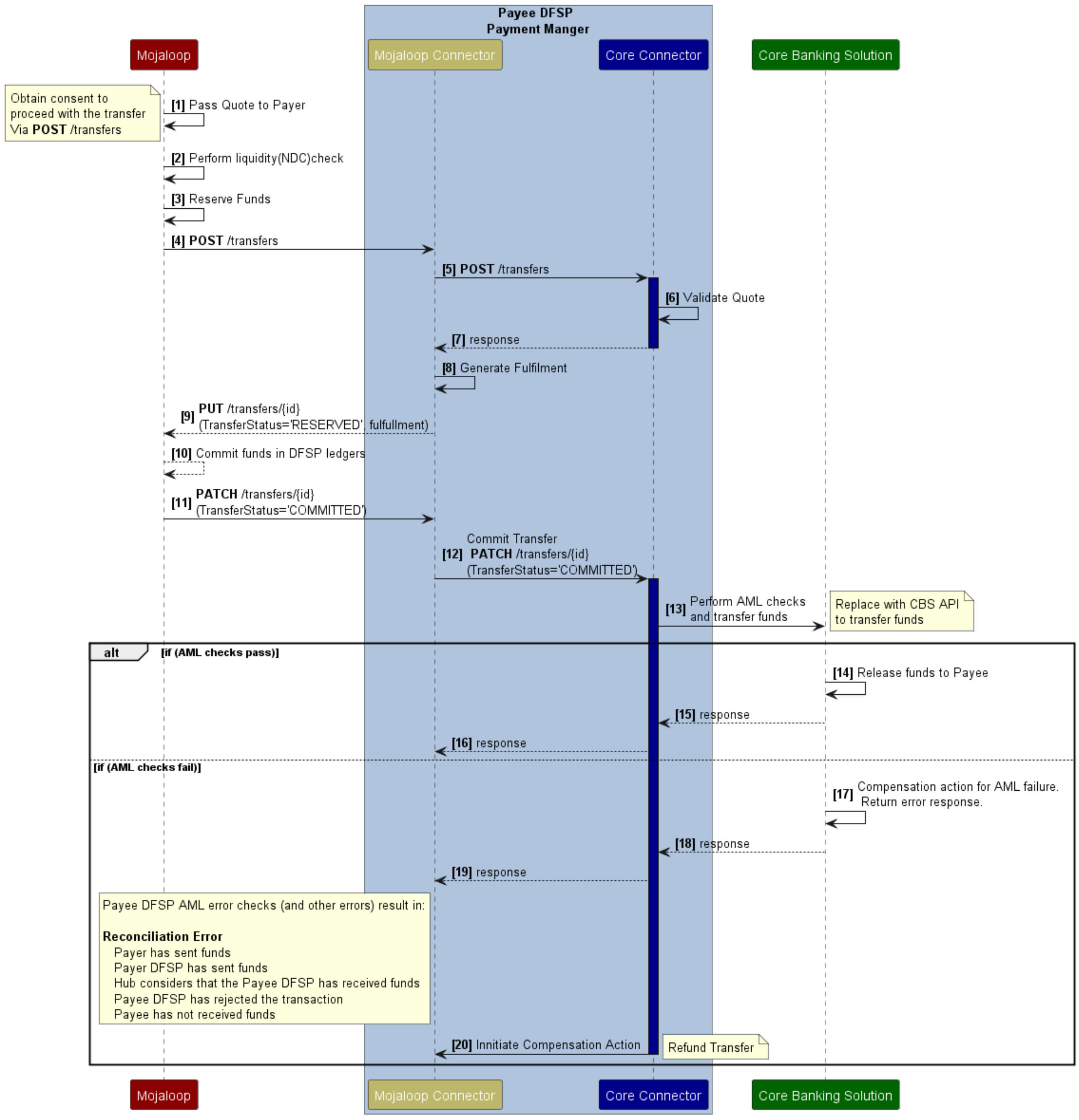
The ideal pattern for completing a transfer as Payee DFSP includes a reservation step that aligns with the Transfer phase, and then a commit phase that aligns with the notification that the transfer has been completed. Please see the details below.



Payee DFSP
Payment Manger
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution
Obtain consent to proceed with the transfer Via POST /transfers
[1] Pass Quote to Payer DFSP
[2] Perform liquidity(NDC)check
[3] Reserve Funds
[4] POST /transfers
[5] Create & Reserve Transfer POST /transfers
[6] Reserve float funds
Replace with CBS API to reserve funds in the float account from which the funds will be moved to the Payees account
[7] response (homeTransactionId)
[8] response (homeTransactionId)
[9] Generate Fulfilment
[10] PUT /transfers/{id} (TransferStatus='RESERVED', fulfullment)
[11] Commit funds in DFSP ledgers
[12] PATCH /transfers/{id} (TransferStatus='COMMITTED')
[13] Commit Transfer PATCH /transfers/{id} (TransferStatus='COMMITTED', homeTransactionId)
[14] Commit funds
Replace with CBS API to commit funds transfer funds to Payee account
[15] Release funds to Payee
[16] response
[17] response

Payee DFSP Single integration on PATCH notification

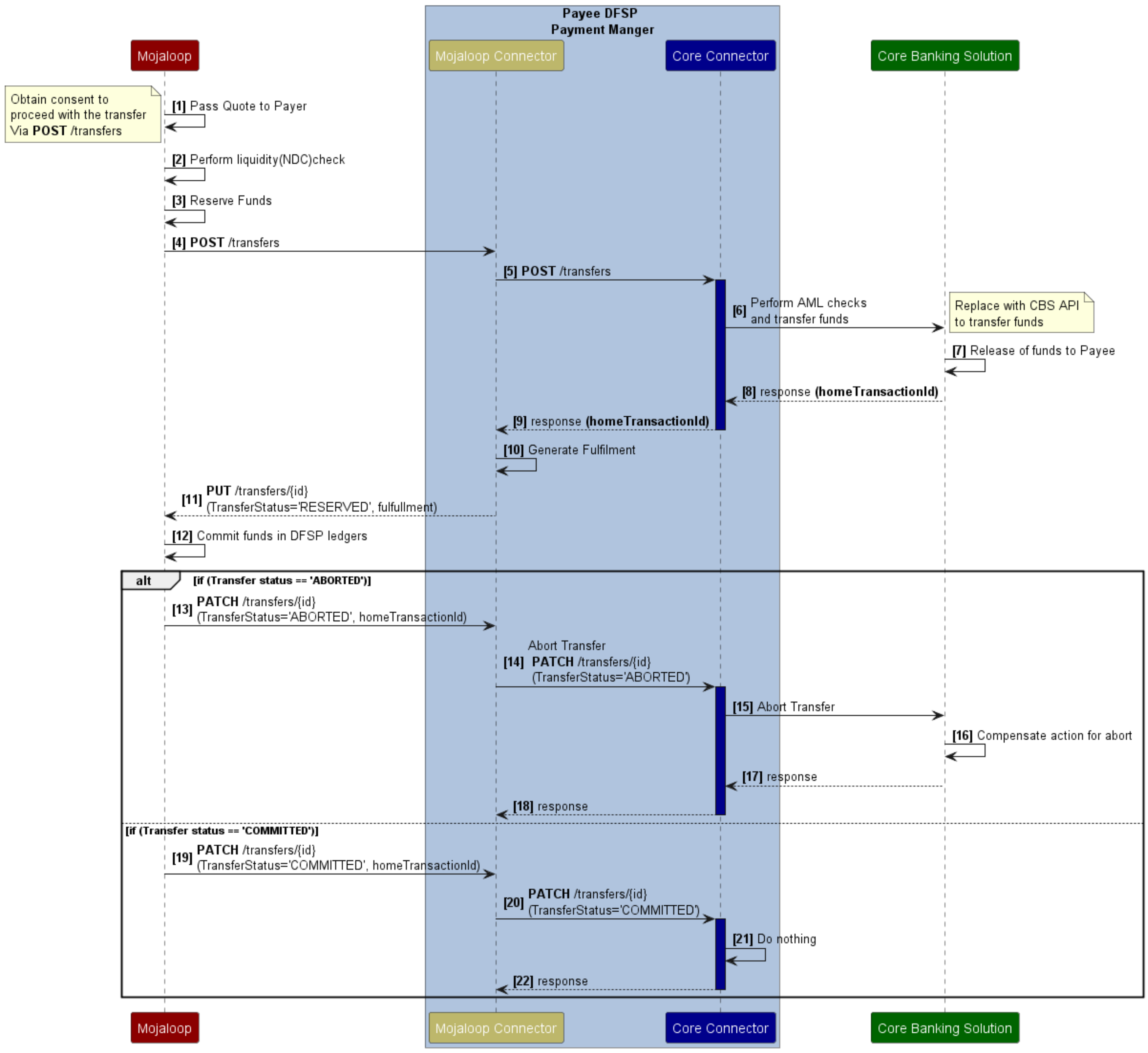
If the DFSPs core banking system only supports a single transfer integration, then this pattern is the preferred integration. The approach only initiates the transfer on the patch notification after the transfer is committed in Mojaloop. This approach has the benefit of only releasing funds after the transaction has been completed and the position ledgers in Mojaloop set to fund the transfer. The disadvantage of this integration pattern is when executing the transfer in the core-banking system fails. If this integration approach is chosen, then effort should go into designing the mitigation steps that must be made if the transfer fails, in order to prevent reconciliation errors.


Payee DFSP
Payment Manger
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution
Obtain consent to proceed with the transfer Via POST /transfers
[1] Pass Quote to Payer
[2] Perform liquidity(NDC)check
[3] Reserve Funds
[4] POST /transfers
[5] POST /transfers
[6] Validate Quote
[7] response
[8] Generate Fulfilment
[9] PUT /transfers/{id} (TransferStatus='RESERVED', fulfullment)
[10] Commit funds in DFSP ledgers
[11] PATCH /transfers/{id} (TransferStatus='COMMITTED')
[12] Commit Transfer PATCH /transfers/{id} (TransferStatus='COMMITTED')
[13] Perform AML checks and transfer funds
Replace with CBS API to transfer funds
alt
[if (AML checks pass)]
[14] Release funds to Payee
[15] response
[16] response
[if (AML checks fail)]
[17] Compensation action for AML failure. Return error response.
[18] response
[19] response
Payee DFSP AML error checks (and other errors) result in:
Reconciliation Error
Payer has sent funds
Payer DFSP has sent funds
Hub considers that the Payee DFSP has received funds
Payee DFSP has rejected the transaction
Payee has not received funds
[20] Innitiate Compensation Action
Refund Transfer

Anti-pattern: Payee DFSP Single integration on transfer

This pattern is considered an anti-pattern and where possible avoided. This design performs the single interaction during the transfer phase. If there is an error and the transfer aborts, then compensation action of the aborted transfer is made in the core banking system. The reason why this is not recommended, is because this transfer clears immediately, it is not possible to guarantee success of the compensation action.

There is one use case where this integration pattern may be used, and that is if the account involved is a loan and there is no release of funds but rather an acknowledgement of a debt payment.


Payee DFSP
Payment Manger
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution
Obtain consent to proceed with the transfer Via POST /transfers
[1] Pass Quote to Payer
[2] Perform liquidity(NDC)check
[3] Reserve Funds
[4] POST /transfers
[5] POST /transfers
[6] Perform AML checks and transfer funds
Replace with CBS API to transfer funds
[7] Release of funds to Payee
[8] response (homeTransactionId)
[9] response (homeTransactionId)
[10] Generate Fulfilment
[11] PUT /transfers/{id} (TransferStatus='RESERVED', fulfullment)
[12] Commit funds in DFSP ledgers
alt
[if (Transfer status == 'ABORTED')]
[13] PATCH /transfers/{id} (TransferStatus='ABORTED', homeTransactionId)
Abort Transfer
[14] PATCH /transfers/{id} (TransferStatus='ABORTED')
[15] Abort Transfer
[16] Compensate action for abort
[17] response
[18] response
[if (Transfer status == 'COMMITTED')]
[19] PATCH /transfers/{id} (TransferStatus='COMMITTED', homeTransactionId)
[20] PATCH /transfers/{id} (TransferStatus='COMMITTED')
[21] Do nothing
[22] response
Mojaloop
Mojaloop Connector
Core Connector
Core Banking Solution