|  |
| --- |
| Exception Handling and Reconciliation |
| *Open API for FSP Interoperability Specification* |

**Table of Contents**

[1 Preface 2](#_Toc517962272)

[1.1 Conventions Used in This Document 3](#_Toc517962273)

[1.2 Document Version Information 4](#_Toc517962274)

[2 Introduction 5](#_Toc517962275)

[3 Multi-hop Interledger Payments. 6](#_Toc517962276)

[4 Timeouts 6](#_Toc517962277)

[4.2 Communication timeout 8](#_Toc517962278)

[5 Error Cases 9](#_Toc517962279)

[5.1 Request from Payer DFSP to IST 10](#_Toc517962280)

[5.2 Request from IST to Payee DFSP 11](#_Toc517962281)

[5.3 Response from Payee DFSP to IST 12](#_Toc517962282)

[5.4 Status update from IST to Payee DFSP 13](#_Toc517962283)

[5.5 Response from IST to Payer DFSP 14](#_Toc517962284)

[6 Applied to Cash-out use case 15](#_Toc517962285)

# Preface

This section contains information about how to use this document.

## Conventions Used in This Document

The following conventions are used in this document to identify the specified types of information

|  |  |  |
| --- | --- | --- |
| Type of Information | Convention | Example |
| Elements of the API, such at resources | Boldface | **/authorization** |
| Variables | Italicswitin angle brackets | *<ID>* |
| Glossary terms | Italics on first occurence; defined in *Glossary* | The purpose of the API is to enable interoperable financial transactions between a *Payer* (a payer of electronic funds in a payment transaction) located in one *FSP* (an entity that provides a digital financial service to an end user) and a *Payee* (a recipient of electronic funds in a payment transaction) located in another FSP. |
| Library documents | Italics | User information should, in general, not be used by API deployments; the security measures detailed in *API Signature* and *API Encryption* should be used instead. |

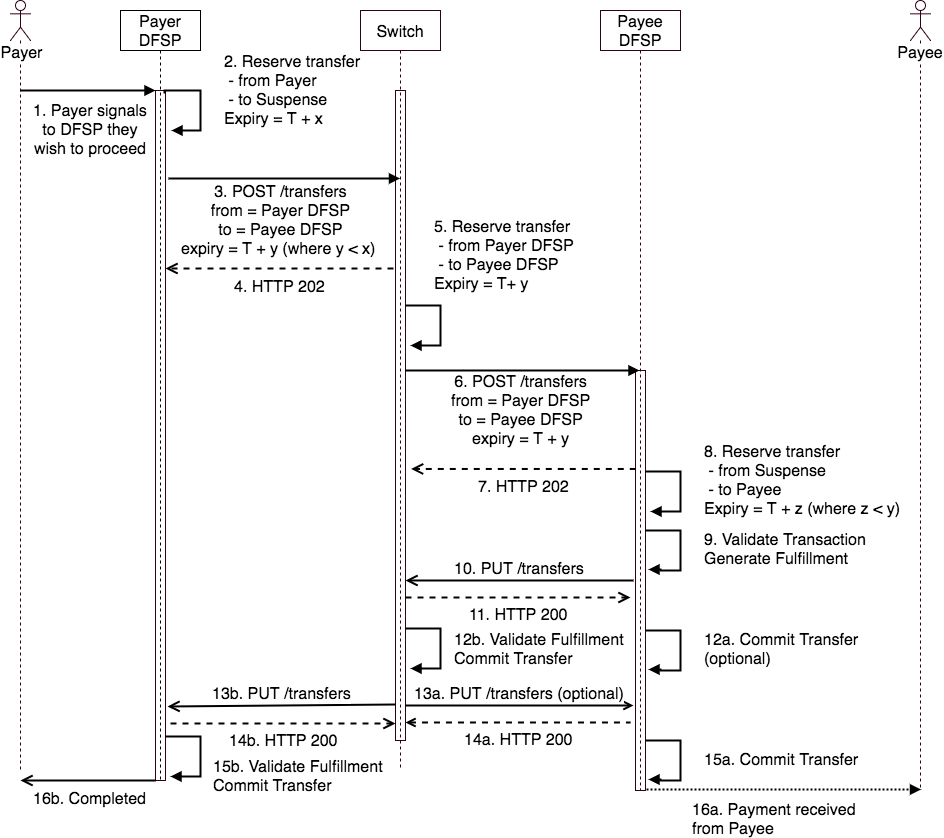
## Document Version Information

|  |  |  |
| --- | --- | --- |
| Version | Date | Change Description |
| 1.0 | 2018-03-13 | Initial version |

# Introduction

This document describes error cases that create a need for reconciliation between different systems. However, it does not take into account how likely it is that these cases may occur. If it is possible that systems may have different statuses for the same transaction, then there must be a process for how such situations may be mitigated.

It is only the **transfer** service that is moving funds, so any transactions that have been done prior to the first transfer from the payer do not have any impact on ledger balances. It is only error cases in the transfers service that can cause a need for reconciliation. Standard transfers flow is as shown in the following image.



Each system (Payer FSP, Switch, Payee FSP) will record the status of the transaction in its own database at the time of commit or rollback. The need for reconciliation happens only when there is a possibility that the status of the same financial transaction is different in different databases.

Note that the Payee DFSP will elect to commit the transfer either upon receipt of the acknowledgement from the IST that it has got the fulfillment (Step 12a) or upon notification by the IST that it has committed the transfer (Step 15a).

# Multi-hop Interledger Payments

This flow is a variation on that of the typical Interledger protocol implementation. It is only possible when the Payee FSP is fulfilling two roles. In this scenario, the Payee FSP is the connector between the *Interoperability Service for Transfer* (IST) and the Payee DFSP ledger, and also the agent of the receiver.

This means the Payee FSP generates the fulfillment (done by the receiver in an Interledger payment) but also assumes the risk inherent in connecting two ledgers (the IST ledger and internal Payee FSP ledger).

Therefore, instead of generating the fulfillment and immediately committing the transfer on the local ledger, the FSP can withhold the fulfillment from the local ledger until it confirms that the fulfillment has been delivered to the IST or even until it has confirmation from the IST that it has been delivered to the Payer DFSP. This mitigates the risk of releasing the funds to the payee and then being unable to deliver the fulfillment to the IST.

The determination of when to release the funds to the payee is risk-based and could vary based on a variety of contextual variables, such as the amount or transaction type. It is also possible that the scheme rules in a deployment will prescribe how and when this must be done.

In a scenario where the Payee FSP is providing payment services to a third party (for example, a merchant) and that third party can process the incoming payment messages and generate fulfillments in real-time, the Payee FSP is no longer in the role of receiver. Therefore, it must follow the correct protocol behavior and commit the local transfer as soon as it receives the fulfillment from the third party.

# Timeouts

An important part of the protocol for transfers is the expiry of both the transfer reservation and the message delivery, and how these are configured in a system and documented in the scheme rules and SLAs.

### Reservation timeout

The Open API defines a timeout for a transfer reservation. This is a business rule configured on each system and should be guided by scheme rules. When a system such as the Payer FSP is creating a reservation, this is done with a specified timeout. After this time, the reservation will be rolled-back. The rollback will happen even if no callback is received on the response to the transfers service.

When this timeout happens, there is a possibility that the Payee FSP and Switch have committed the transaction and the Payer FSP has performed a rollback and returned the funds to the Payee.

Without this automatic expiry, there would not be the same need for reconciliation as reservation would not be released without knowing the status of the transaction. However, without a rollback it is impossible to synchronize the commit across multiple distributed ledgers without the risk of money being locked up indefinitely.

The Interledger protocol defines an intentional ordering of commits, starting at the receiver and cascading backwards. The expiry on each system should be configured such that it telescopes down as the transfers are made further from the payer system. In this way, any two systems can focus solely on synchronized state between themselves following which the system closest to the Payer begins to focus on synchronizing state with the next system.

The only time a reconciliation is required is when two systems are unable to reconcile state before the expiry on one of the systems. The protocol ensures that this will not have a negative financial impact on the Payer and Payee and so the scheme rules will always favor the customer over the scheme participants.

## Communication timeout

*Communication timeout* is different from the reservation timeout. Communication timeout is used to trigger a resend of a transfer or a commit if a system has not responded to a HTTP call such as **POST /transfers** or **PUT /transfers**. The communication retry is used to try to get the status of a transfer before the transfer reservation timeouts.

Communication timeout is not related to the need for reconciliation. Rather, the communication timeout and number of subsequent retries should be configured, such that the maximum number of attempts is made to synchronize state before the reservation timeout.

### SLAs and Incentives

Due to the nature of the protocol, there is a financial incentive to pass on the commit trigger (fulfillment) after processing it. For example, if the Payee DFSP sends the **/transfers** call back to the IST and gets no response, it is incented to retry before the expiry, otherwise the transfer at the IST in its favor will be rolled back.

In the case of a shared IST service that is not a financial participant, there is no financial incentive. Rather, the SLAs of the scheme should be structured in such a way that the IST takes on some financial liability if it does not pass the commit trigger to the Payer DFSP in time.

# Error Cases

There are different types of error cases that could cause the systems to get different state regarding the same the transaction. Each error case refers to a break-down of communication at some point in the flow of messages as defined in the image above.

## Request from Payer DFSP to IST

This occurs in Steps 3 or 4.

### Risk:

In this case, the Payer DFSP has reserved funds from the Payer’s account, but is unsure if the transfer request was received by the IST or not.

### Mitigation:

The **/transfers** API call is idempotent, so the Payer DFSP should retry Step 3 until:

* Step 4 completes successfully *or*
* It receives the request from the IST in Step 13 *or*
* It receives a settlement message indicting that the payment has been cleared at the IST *or*
* The time remaining before the reservation timeout at the Payer DFSP approaches the requested expiry in the transfer request to the IST.

### Reconciliation:

The Payer DFSP will be able to reconcile its position with the rest of the network following the next settlement cycle. It should configure the reservation timeout to be greater than the period of the settlement cycle.

If the transaction appears in the settlement report, then the Payer DFSP can commit the local transfer. If not, it can assume that the transfer was not done at the IST and therefore it can roll back.

## Request from IST to Payee DFSP

This occurs in Steps 6 or 7.

### Risk

In this case, the Payer DFSP has reserved funds from the Payer’s account and the IST has reserved funds from the Payer DFSP.

The Payer DFSP has liquidity locked up until this transaction completes.

### Mitigation

The **/transfers** API call is idempotent, so the IST should retry Step 6 until:

* Step 7 completes successfully *or*
* It receives the request from the Payee DFSP in Step 10
* The time remaining before the reservation timeout at the IST approaches the requested expiry in the transfer request to the Payee DFSP.

### Reconciliation

During the settlement cycles the IST must only include transfers that have been committed.

It must assume that if it doesn’t send a transfer response to the Payer DFSP that it will eventually roll back the transfer on its ledger.

It must also assume that the Payee DFSP, if it received the transfer request, will be making every effort to return the response (Step 10) as this is the only way the Payee DFSP can be certain it will be settled for the transaction.

## Response from Payee DFSP to IST

This occurs in Steps 10 or 11.

### Risk

In this case, the Payee DFSP has reserved the funds to pay the Payee, but needs to ensure that the IST has received the commit message before they roll back.

### Mitigation

The **/transfers** API call is idempotent so the Payee DFSP should retry Step 10 until:

* Step 11 completes successfully *or*
* The expiry (T + z) for the reservation made at Step 8 is exceeded *or*
* It receives a settlement message indicting that the payment has been cleared at the IST.

### Reconciliation

The Payee DFSP rolled back the transfer, but the transaction is present in the settlement reports.

The scheme rules will determine how to handle this situation, both within the window before the Payer DFSP rolls back their transfer (that is, while the payer funds are still reserved) and after.

One possibility is for the Payee DFSP to roll the transfer back, but to hold the reserved funds in a separate holding account (that is, not return it to working capital) until the result of the transaction at the IST is known.

## Status update from IST to Payee DFSP

This occurs in Steps 13a or 14a.

### Risk

In this case, the Payee DFSP has not paid the Payee, and the IST has committed the transfer from the Payer DFSP to the Payee DFSP.

### Mitigation

The **/transfers** API call is idempotent, so the IST should retry Step 13a until:

* Step 14a completes successfully *or*
* It includes the transaction in a settlement report sent to the Payee DFSP.

### Reconciliation

If the Payee DFSP has held back a payment pending the confirmation from the IST, then there are no funds at risk; however, the payee will be waiting for their payment.

The IST and Payee DFSP can reconcile based on the next settlement report, or the Payee DFSP can query for the status of the transfer by repeating Step 10.

## Response from IST to Payer DFSP

This occurs in Steps 13b or 14b.

### Risk

In this case, the Payee DFSP has paid the Payee, and the IST has committed the transfer from the Payer DFSP to the Payee DFSP.

### Mitigation

The **/transfers** API call is idempotent, so the IST should retry Step 13b until:

* Step 14b completes successfully *or*
* It includes the transaction in a settlement report sent to the Payer DFSP.

### Reconciliation

During the settlement cycles the IST must only include transfers that have been committed.

It must assume that, if it doesn’t send a transfer response to the Payer DFSP, the Payer DFSP will eventually roll back the transfer on its ledger.

It must also assume that, if it produces a settlement report that includes this transaction and this is sent to the Payer DFSP, then the Payer DFSP will commit the transfer.

As there is no financial incentive for the IST to ensure delivery of the transfer response, the scheme rules should ensure that there is an appropriate SLA and penalty in place for this.

The Payer DFSP is at risk if it rolls back the transfer on its own ledger when the transfer at the IST has been committed; therefore the Payer DFSP may proactively get the commit message by doing an enquiry at the IST using **GET /transfers**. The IST should not depend on this behavior.

# Applied to Cash-out use case

The Cash-Out use case is a high-risk case, in which it is likely that the Payee DFSP will wait for confirmation from the IST that the fulfillment has been delivered to the Payer DFSP before committing the funds.

If the Response from Payee DFSP to IST error case is applied to this use case, then the following could happen:

1. End user goes to an Agent to do a Cash-out.
2. Agent (Payee) will enter all transaction details and send to the Payer FSP as a request to transfer.
3. Payer FSP will reserve the amount (Step 2) on the End users account and send **POST /transfers** to the switch (Step 3).
4. Switch will reserve (Step 5)and forward the **POST /transfers** (Step 6) to the Payee FSP
5. Payee FSP will generate the fulfillment (Step 9).
6. Payee FSP will send a callback to the switch (Step 9), but will *not* receive response (Step 11); for example, due to network failure.
7. Payee FSP will retry, but continue to fail.
8. Payee FSP will send a notification to the Agent that the cash-out has not failed, but requires manual intervention.

Either of the following flows will then occur.

**A: Switch never receives the commit message (Step 10)**

1. Switch will timeout when the reservation timer is passed as no callback is received from the Payee FSP.
2. Switch will send an error callback to the Payer FSP (Step 13b)that transaction timed out.
3. Payer FSP will release the money back to the end user and send a notification that the cash-out failed.
4. Switch will send a status update to the Payee FSP (Step 13a) indicating that the transfer has been rolled back.
5. If the status update is not delivered the Payee FSP will reconcile with the Switch following the next settlement cycle and note that the transaction was rolled back.
6. Payee FSP will notify the agent that the transaction failed(Step 16a).

**B: Payee DFSP never receives the commit response/ack (Step 11)**

1. Switch will receive the commit message and pass this on to the Payer FSP and the Payer is notified that the transaction is successful (Steps 13 – 16).
2. Payee FSP never receives the acknowledgement and times out.
3. Payee FSP rolls back the transfer and puts the funds aside into a temporary holding account.
4. Either the Payee FSP will receive the transfer status update from the Switch(Step 13a) or will reconcile with the Switch following a settlement cycle and note that the transaction was completed.
5. Payee FSP will transfer funds on hold to agent and notify the agent that the transaction was successful (Step 16a).

In scenario B the scheme rules must dictate how the agent is expected to deal with the customer. The likely scenario is that the agent has the facility to log a dispute with the Payee FSP.

The outcome may be a dispute ticket number that the payer can use to request a refund of their payment via their own FSP or to collect the cash from the agent later.

Also, the likelihood of scenario B occurring is very low, as this would imply that the connection from the IST to the Payee FSP dropped at the instant between Steps 10 and 11. There should be no processing at the IST between these two steps other than to store the incoming commit message.