EUROPEAN CENTRAL BANK
EUROSYSTEM

BANK OF JAPAN

STELLA – joint research project of the European Central Bank and the Bank of Japan

Synchronised cross-border payments

June 2019

Contents

1 Introduction 1

2 Related analysis and main results of Stella phase 3 3

2.1 Related analysis around ledger interoperability 3

2.1.1 Research by central banks 3

2.1.2 Selected private sector initiatives 4

2.2 Main results of the joint analysis of Stella phase 3 5

3 A protocol for interledger payments 7

4 Payment methods 12

4.1 Trustline 13

4.2 On-ledger escrow using HTLC 15

4.3 Third party escrow 16

4.4 Payment channels 16

4.4.1 Simple payment channel 17

4.4.2 Conditional payment channel with HTLC 18

4.5 Ledger requirements for payment methods 19

4.6 Summary 19

5 Experimentation 20

5.1 Overview of experiments 20

5.2 Experiment with centralised ledgers 21

5.3 Experiments with DLT ledgers 22

5.3.1 Experiment without ILP 22

5.3.2 Experiment with ILP 23

5.4 Cross-platform experiment 23

5.5 Results of experiments 24

6 Assessment of payment methods 24

6.1 Safety 24

6.2 Liquidity efficiency 28

6.3 Summary 28

7 Additional considerations 29

7.1 Safety of individual payments and atomicity of the payment chain 29

7.2 Impact of ledger processing speed and operational availability 30

7.3 The free option problem 31

Annex 1: Process flows and ledger requirements 33

Annex 2: Details of experiments 45

EUROPEAN CENTRAL BANK
EUROSYSTEM

STELLA

BANK OF JAPAN

Project Stella

Synchronised cross-border payments

1 Introduction

The emergence of distributed ledger technologies (DLT) in recent years has spurred discussions around the future of financial market infrastructures (FMI) supporting payments and securities settlement[1]. Project Stella is a joint research undertaking by the European Central Bank (ECB) and the Bank of Japan (BOJ), launched in December 2016, which aims to contribute to the ongoing debate with experimental work and conceptual studies exploring DLT's opportunities and challenges for FMI. The current report (in the following referred to as Stella phase 3) builds on the insights gained from the previous two phases which had focused on the processing of large-value payments using DLT (September 2017)[2] and securities delivery versus payment (DVP) in a DLT environment (March 2018).[3] Stella phase 3 explores innovative solutions for cross-border payments, i.e. payments between currency areas.[4]

Cross-border payments involve various entities across multiple jurisdictions. While the demand for cross-border payments is on the rise, they are often characterised as slow and costly when compared with domestic payments.[5] In particular the lack of common communication or messaging standards across systems often hinders seamless interoperability. While initiatives exist to address these current

[1] See *Payment and Settlement Systems Report*, Bank of Japan, March 2019 for a list of research projects conducted by central banks.

[2] *Payment systems: liquidity saving mechanisms in a distributed ledger environment*, ECB and BOJ, September 2017.

[3] *Securities settlement systems: delivery-versus-payment in a distributed ledger environment*, ECB and BOJ, March 2018.

[4] The joint research was conducted by Dirk Bullmann (ECB team leader), Andrej Bachmann, Giuseppe Galano, Josip Kenjeric and Cedric Humbert, with contributions from Anna Kearney and Diego Castejon Molina, from the ECB (Directorate General Market Infrastructure and Payments and Directorate General Information Systems); and by Michinobu Kishi (BOJ team leader), Norio Hida, Hidetaka Enomoto, Toshio Okuji, Tetsuro Matsushima and Amika Matsui from the BOJ, with contributions from Shuji Kobayakawa (Professor at Meiji University and Advisor to the BOJ Stella team).

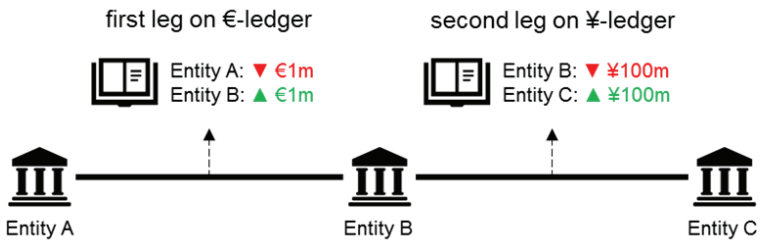
[5] See *Cross-border retail payments*, Committee on Payments and Market Infrastructures (CPMI), February 2018, and *Correspondent banking*, CPMI, July 2016.

inefficiencies of cross-border payments[6], the safety aspects of transactions across payment ledgers[7] remain a challenge.

As Figure 1 illustrates, credit risk might arise if a party fails prior to completion of a cross-border transfer. In this simplified example, Entity A intends to send ¥100 million to Entity C by sending €1 million to Entity B[8] (e.g. an intermediary bank), which has access to both euro and yen ledgers, and in turn sends ¥100 million on behalf of Entity A. If Entity B fails after the first leg of the transfer is complete (i.e. the €1m transfer from A to B) but before the second leg of the transfer is complete, Entity A faces risk of loss of its funds. This risk can be mitigated if the payments are synchronised and funds are locked, but in today's world such synchronisation rarely happens.

Figure 1

Simplified example of credit risk arising from cross-border funds transfer



Against this background, Stella phase 3 explores whether cross-border payments could potentially be improved, especially in terms of safety by using new technologies. Concretely, it analyses global interoperability on the basis of a protocol for interledger payments being used (i) between a centralised ledger (e.g. a ledger operated by commercial banks or a real-time gross settlement (RTGS) system operated by central banks) and a DLT ledger, (ii) between DLT ledgers, and also (iii) between centralised ledgers. It thereby focuses on back-end arrangements of cross-border fund transfers (see Figure 2).

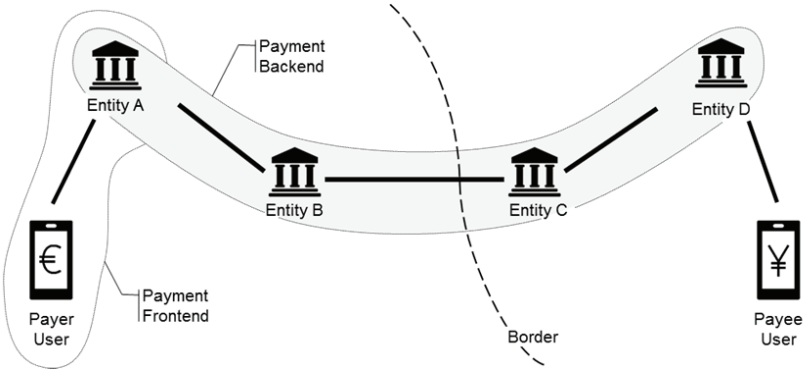
The analysis and experimental results presented in this report are not geared towards replacing or complementing existing arrangements, which include central bank-operated payment systems. Moreover, legal and regulatory aspects are outside the scope of the project. Project Stella offers a contribution to a wider debate on the possible usage of DLT in the field of payments and financial market infrastructure services.

[6] For example, by achieving common communication standards through the adoption of ISO 20022.

[7] Cross-border payments involve a variety of factors which may act as hindrances to safe and efficient transfers, such as discrepancies in legal and regulatory standards between jurisdictions, implications of the use of multiple currencies, and the involvement of multiple ledgers. This study primarily focuses on the facet of the involvement of multiple ledgers.

[8] Entity B would conduct currency conversion (€1 = ¥100).

Figure 2
Stylised image of cross-border payment arrangements



Chapter 2 summarises main results of the study in view of already existing analysis around potential DLT usage. Chapter 3 outlines the concept of a protocol for interledger payments and details of how the protocol works. Chapter 4 explains the relevant payment methods and lists ledger functionalities that are required for individual payment methods. Chapter 5 summarises the experimentation conducted by BOJ and ECB engineers using the protocol. Chapter 6 assesses the safety and efficiency aspects of specific payment methods based on the utilised protocol. Chapter 7 offers additional considerations beyond safety and efficiency.

2 Related analysis and main results of Stella phase 3

2.1 Related analysis around ledger interoperability

Several public and private sector initiatives explore innovative solutions aimed at enhancing interoperability between different ledgers and enabling synchronised settlement of cross-border payments. The following subsections give an overview of selected initiatives.

2.1.1 Research by central banks

In the context of analysis around the renewal of their RTGS service, Bank of England explored wider interoperability, including synchronised settlement of RTGS payments with payments in other systems.[9] In that context, a Proof of Concept was conducted on synchronised settlement of payments on two different ledgers under a high-value

[9] *A blueprint for a new RTGS service for the United Kingdom*, Bank of England, May 2017.

cross-border payment scenario.[10] Bank of England also laid out a potential model for synchronised payment settlement which involves a trusted third party that offers synchronisation services which could be applied to simultaneous settlement across ledgers and currencies.[11]

Bank of Canada and the Monetary Authority of Singapore explored interoperability between two DLT platforms using Hashed Timelock Contracts (HTLC[12]), which were also studied in the second phase of Project Stella.[13] They simulated a cross-border high-value transfer across DLT-based RTGS systems on different platforms.

2.1.2 Selected private sector initiatives

A community group of the World Wide Web Consortium (W3C) is developing further the Interledger Protocol (ILP) – a set of rules that allows payments to be sent across different types of ledgers.[14] ILP is built on the initial proposal laid out in the whitepaper "A Protocol for Interledger Payments" (Thomas and Schwartz, 2015).[15] Chapter 3 provides a more detailed description of the underlying concepts of ILP.[16]

Ripple has developed xCurrent, which connects financial institutions via a global network of participating entities (RippleNet). xCurrent is built around ILP and enables bidirectional communication between participating entities and coordination of payments across ledgers.[17]

SWIFT established a new standard for participating institutions – SWIFT gpi (global payments innovation) – to improve speed, security and transparency in cross-border payments across the correspondent banking network. There are three key features to SWIFT gpi: (i) an end-to-end payments tracking database that allows real-time monitoring of the payment status; (ii) a directory that provides operational information such as that on participating institutions, currencies and cut-off times to be able to find the best payment route; and (iii) a central service that provides participating institutions with a global view of other members' adherence to newly created set of rules to enhance business practices.[18]

---

[10] *FinTech Accelerator Proof of Concept: exploring the synchronised settlement of payments using the Interledger Protocol*, Bank of England, July 2017.

[11] *Call for interest: Synchronised settlement in central bank money*, Bank of England, August 2018.

[12] For the details of HTLC, see Section 4.2 and Annex 1.

[13] *Enabling Cross-Border High Value Transfer Using Distributed Ledger Technologies*, Bank of Canada and Monetary Authority of Singapore, May 2019.

[14] https://www.w3.org/community/interledger/

[15] https://interledger.org/interledger.pdf

[16] Ripple conducted a public experiment of ILP at an Interledger workshop held in June 2017.

[17] https://ripple.com/files/ripple_solutions_guide.pdf

[18] https://www.swift.com/resource/swift-gpi-brochure

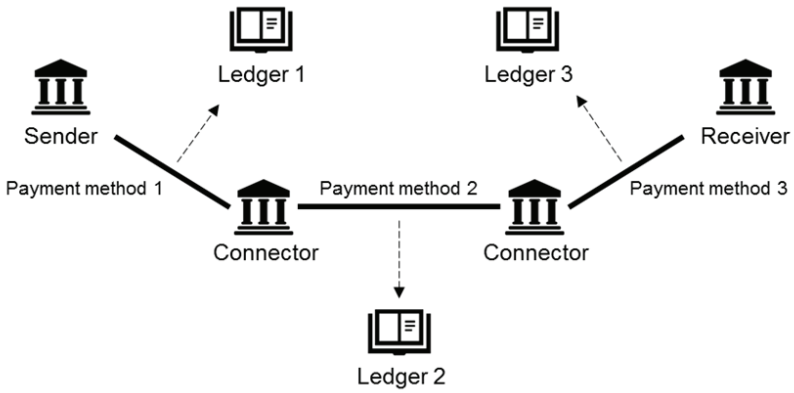
2.2 Main results of the joint analysis of Stella phase 3

Stella phase 3 explores innovative solutions to improve cross-border payments (see stylised payment chain in Figure 3) especially in terms of safety. It builds on the experimental and conceptual work previously conducted under the umbrella of Project Stella and also takes into consideration research work of other central banks and private sector entities (see Section 2.1).

The second phase of Project Stella[19] identified a new approach for settlement across ledgers through HTLC that could potentially allow the mitigation of credit risks through the synchronisation of settlement.

Stella phase 3 expands the scope of this analysis. It studies a protocol introduced in the afore-mentioned whitepaper "A Protocol for Interledger Payments", which attempts to present a ledger-agnostic protocol that synchronises payments across different types of ledgers.[20] It also assesses the safety and efficiency implications of a variety of payment methods which could be used in the cross-ledger payment.

Figure 3
Stylised payment chain constituting a cross-border payment



These payment methods are:

1. **Trustline** is an arrangement between the payer and the payee outside the ledger where the payer promises to make a payment if the payee fulfils a predefined condition. At the same time, the total of payments which has not been settled must not exceed the predetermined maximum amount that the payer can pay without settlement on the ledger.

[19] *Securities settlement systems: delivery-versus-payment in a distributed ledger environment*, ECB and BOJ, March 2018.

[20] Applicability to DLT ledgers and centralised ledgers was confirmed through a series of experiments as further described in Chapter 5.

2. **On-ledger holds/escrow using HTLC (hereafter "on-ledger escrow")** allow conditional transfers which are recorded on the ledger and enforced by the ledger if the payee fulfils a predefined condition.

3. **Third party escrow** is conceptually similar to the on-ledger escrow but relies on a third party which is trusted by the payer and the payee rather than on the ledger to enforce the conditional transfers.

4. **Simple payment channel** is an arrangement between the payer and payee using escrowed funds in a shared temporary account on the ledger. Both parties promise to exchange signed claims off-ledger, which represents their entitlement to a specific portion of escrowed funds, if the payee fulfils a predefined condition. Only the final net position of multiple bilateral payments is actually settled on the ledger.

5. **Conditional payment channel with HTLC (hereafter "conditional payment channel")** is similar to simple payment channel in the sense that both parties exchange signed claims off-ledger, but in addition has an enforcement mechanism by the ledger for the transfers based on whether the payee fulfils a predefined condition.

These five possible methods show distinctive characteristics which can be summarised as follows (see also Table 1): (i) whether individual payments are settled on-ledger or recorded off-ledger, (ii) whether funds are locked or escrowed, (iii) whether payments are enforced when the predefined condition for the payment is fulfilled, and (iv) whether specific ledger functionalities are required to conduct transfers, such as the functionalities to enforce conditional transfers and process signed claims.

Table 1
Overview of payment methods and specific ledger requirements

Payment method	On-ledger/ Off-ledger	Escrow/ Lock	Enforcement of conditional payment	Specific ledger requirements
Trustline	Off-ledger	No	No enforcement	No
On-ledger escrow	On-ledger	Yes	Enforced by ledger	Yes
Third party escrow	On-ledger	Yes	Enforced by third party	No
Simple payment channel	Off-ledger	Yes	No enforcement	Yes
Conditional payment channel	Off-ledger	Yes	Enforced by ledger	Yes

In relation to safety, Stella phase 3 concludes that on-ledger escrows, third-party escrows, and conditional payment channels, all of which have enforcement mechanisms, can ensure that each transacting party who completely satisfies its responsibilities in the transaction process is not exposed to the risk of incurring a loss on the principal amount being transferred.

As for liquidity efficiency, the five payment methods considered in this report can be grouped as follows in the order of efficiency – (1) trustline, (2) on-ledger escrow and third party escrow, and (3) simple and conditional payment channels. Trustline appears to be superior to the other payment methods since it is the only post-funded payment method. The liquidity efficiency of on-ledger escrow and third party escrow (which escrows the money needed only for a single payment) is generally better than that of simple payment channel and conditional payment channel (which escrows the money needed for all the payments that will be dealt with in the payment channel).

In conclusion, from a technical perspective, the safety of today's cross-border payments could potentially be improved by using payment methods that synchronise payments and lock funds along the payment chain. It should, however, be noted that further reflections on legal and compliance issues, the maturity of the technology and a cost-benefit analysis would be required before the possible implementation of such new methods could be considered.

3 A protocol for interledger payments

This chapter describes the concepts of a protocol for interledger payments – a ledger-agnostic protocol – that allows the sender to make payments across different types of ledgers. This protocol is built on the proposal for universal[21] interledger payments outlined in the whitepaper "A Protocol for Interledger Payments" (Thomas and Schwartz, 2015) and more recent developments. Currently, the specification based on this protocol (the Interledger Protocol, or ILP) is mainly developed by a W3C community group. The most recent version of ILP (version 4, or ILPv4) is openly available.[22] This chapter describes the general concepts and how they could be applied to a cross-ledger payment scenario.

The building blocks of the protocol are *Participants, Ledgers and Payment Methods*.

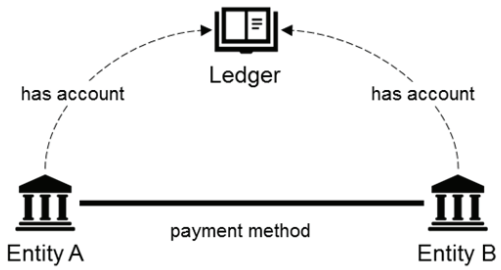
In our context, ledgers are a generic term for any system used to track transfers of value between, and balances on, accounts. Throughout this report, a funds transfer is always recorded on the ledger, while a payment is not always recorded on the ledger because its settlement could be postponed. Participants are entities that have accounts on one or more ledgers and participate in the cross-ledger payment.

[21] In their whitepaper, Thomas and Schwartz (2015) also propose an atomic mode for interledger payments, which is not being considered in this report. The atomic mode has been deprecated and is currently not being developed by the community group of the W3C.

[22] https://interledger.org/rfcs/0027-interledger-protocol-4/

Finally, a payment method is a bilateral agreement between participants on a specific method to make payments and settle obligations on the ledger. The choice of payment method depends on participants' preferences and ledger functionalities.

Figure 4
Stylised setup between two entities



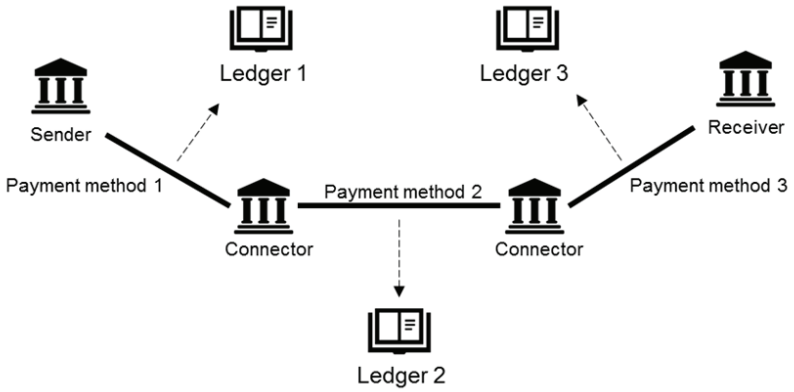
Participants can assume three roles within the Interledger Protocol: that of a *Sender*, *Receiver*, or *Connector*. Connectors are entities with accounts on two or more ledgers who act as liquidity providers that relay payments across ledgers and play a critical role for the successful execution of cross-ledger payments. A liquidity provider enables a cross-ledger payment by exchanging an incoming payment from its account on one ledger for an outgoing payment to its account on another ledger.[23]

Where a single Connector cannot link the payment between the Sender and the Receiver, or cannot do so in an efficient way, multiple Connectors can be composed into a payment chain.[24] Figure 5 shows an example of a payment scenario with three ledgers and two Connectors acting as liquidity providers.

[23] When Connectors relay a payment across ledgers denominated in different currencies, they conduct currency conversion. Connectors are often referred to as liquidity providers in other documentations.

[24] Theoretically, the number of Connectors in a cross-ledger payment is not limited.

Figure 5
Example of a cross-ledger payment chain



All individual payments along the payment chain depend on the fulfilment of the following condition by the payee (the Receiver or Connector(s)): presentation of the *preimage for a cryptographic hash value*[25] before a predetermined time (timeout). The hash value is used to define the payment condition, while the corresponding hash preimage marks the fulfilment of that condition.[26] The same cryptographic hash function,[27] value and preimage must be used in a single cross-ledger payment chain.

Before a cross-ledger payment is initiated, the preimage and its cryptographic hash value are produced by the Receiver, by deriving the hash value from an arbitrary preimage using the hash function. The hash value must then be shared with the Sender together with other terms of the payment, which may include payment amount, payment currency, payment timeout and Receiver information, using external means of communication (e.g. email).[28]

After the initial bilateral Sender-Receiver communication, each individual payment of the cross-ledger payment chain goes through two main phases.

- First, the payer (the Sender or Connector(s)) prepares the payment to the payee (the Receiver or Connector(s)) according to the specific payment method used.[29]

- In the following phase, there are three scenarios that result in the prepared payment being either executed or aborted. If the hash preimage is presented by the payee (the Receiver or the Connector(s)) before the timeout and is verified as correct, the condition for the payment is fulfilled and the payment to the

[25] Hash value is output of a hash function and preimage is the input of it.

[26] The fundamental idea of linking a payment to the presentation of the preimage for a cryptographic hash value before the expiration of a timeout comes from the concept of HTLC, a special type of smart contract that was analysed in Stella phase 2.

[27] The community group developing ILP recommends the use of SHA256.

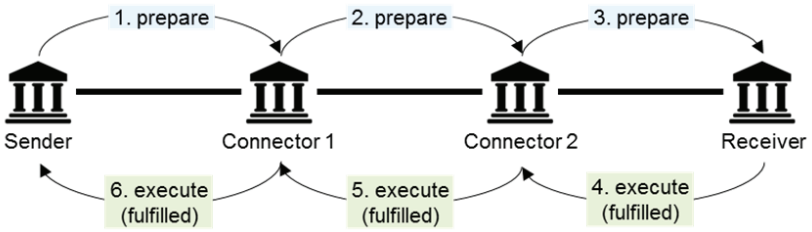
[28] In ILPv4, this could be achieved through a functionality called Simple Payment Setup Protocol.

[29] See Chapter 4 and Annex 1 for details.

payee is executed (fulfilment scenario). Alternatively, if the timeout expires without the correct preimage being presented, the payment is aborted (timeout scenario), or if the payee does not accept the payment, the payment could be aborted even before the expiration of the timeout (reject scenario).[30]

Figure 6 shows a successful payment scenario with two Connectors and illustrates the order of payment preparations and executions. The Sender and Connectors 1 and 2 prepare the payment to Connector 1, to Connector 2, and then to the Receiver, respectively, in this order. Then, the Receiver and both Connectors fulfil the payment conditions by presenting the hash preimage before the timeout. Depending on the arrangement between the Sender and Receiver, possession of the preimage by the Sender could be regarded as evidence of the receipt of payment by the Receiver.

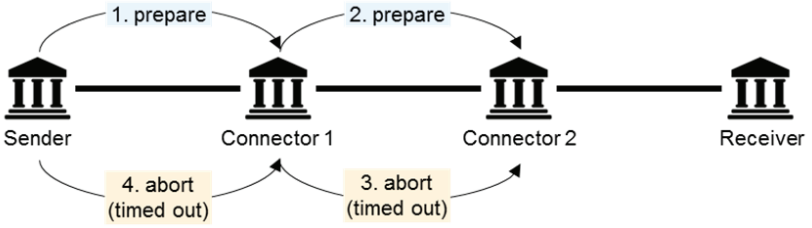
Figure 6
Execution of payments along a payment chain



Note that the arrows signify actions by the originating entities. Also, for simplicity, ledgers are not depicted. The same applies for Figures 7 and 8.

In the scenario of Figure 7, the payment is first prepared but then aborted due to timeout. The Sender and Connector 1 prepare the payment to Connector 1 and Connector 2, respectively, in this order. Then, if Connector 2 remains unresponsive and does not present the hash preimage to Connector 1 before timeout, payments to Connector 2 and Connector 1 are aborted, in this order.

Figure 7
Cross-ledger payment aborted due to timeout

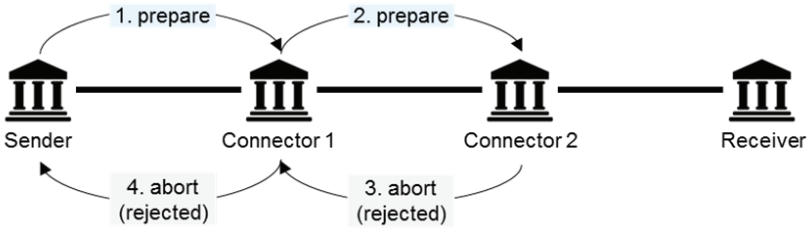


[30] The reject scenario was not part of the initial proposal introduced in the whitepaper, where a payment chain could either be executed or aborted. While it may be possible for the payee to reject a transfer rather than to wait for the timeout, there is no economic incentive for the payee to do so. Nevertheless, it can provide some operational benefits to the participants.

Figure 8 illustrates the payment being prepared and then aborted before the expiration of the timeout due to Connector 2 rejecting the payment. The Sender and Connector 1 prepare the payment to Connector 1 and Connector 2, respectively, in this order. Then, as Connector 2 rejects the payment, the payment to that entity is aborted, which results in payment to Connector 1 also being aborted.

Figure 8

Cross-ledger payment aborted due to rejection



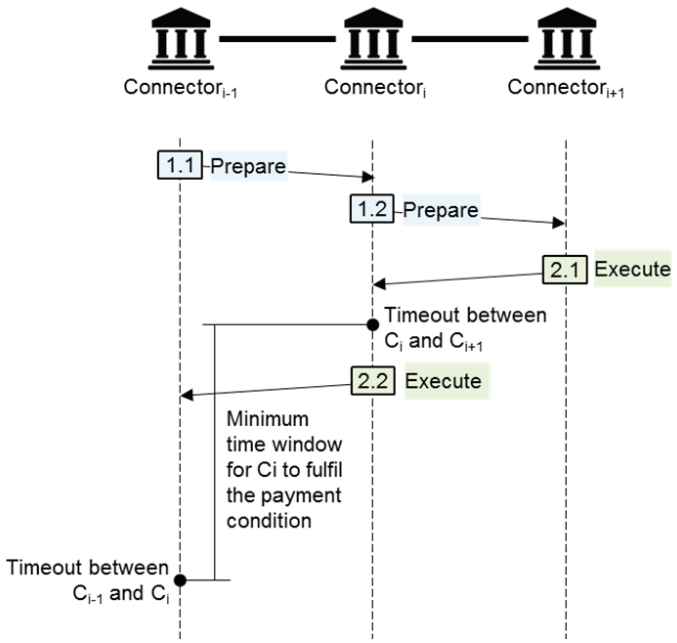
Payment processes based on the protocol require information exchanges between participants as well as with other relevant entities.[31] These include the initial information exchanged between the Sender and the Receiver which does not vary along the payment chain and does not depend on the ledger and the payment method used, as well as other information which varies among each payment in the payment chain (such as fees and timeouts for individual payment conditions). With reference to the latter set of information, it is important to highlight the following considerations:

- The timeout for the fulfilment condition between Connector C_{i+1} (or the Receiver) and C_i should be set reasonably before the timeout for the fulfilment condition between C_i and C_{i-1} (or Sender). This is to allow C_i sufficient time to fulfil the condition after C_{i+1} has fulfilled the condition by presenting the hash preimage received from C_{i+1}. Note that the timeout for the payment to the Receiver can be set at most equal to the timeout agreed upon by the Sender and Receiver and specified in the terms of payment.

- The payment amount between C_{i-1} (or the Sender) and C_i could be different from the amount between C_i and C_{i+1} (or the Receiver), in order to take into account possible fees and denomination in different currencies. Note that the payment amount sent to the Receiver should be at least equal to the payment amount agreed upon by the Sender and Receiver and specified in the terms of payment.

[31] As described in Chapter 4, these include ledgers on which the participants hold accounts and providers of third party escrows.

Figure 9
Concept of timeouts along payment chain



4 Payment methods

This chapter provides an overview and a description of the five payment methods compatible with the concepts of the Interledger Protocol that we consider in this report. The common feature of these payment methods is that the two parties form an agreement that the payee presents a certain secret hash preimage within a certain period of time in exchange for payment. There can be a relatively wide spectrum of such payment methods. The feasibility of each payment method considered in this report depends either on the existence of certain ledger functionalities and/or a trust relationship between the payer and the payee.[32]

This report considers the following payment methods:

- trustlines;
- on-ledger escrow using HTLC;

[32] The names and descriptions of the payment methods are based on Hashed-Timelock Agreements (HTLA) of the Interledger Protocol, as described at https://interledger.org/rfcs/0022-hashed-timelock-agreements/

- third party escrow;[33]
- simple payment channels;
- conditional payment channels with HTLC.

The descriptions of payment methods in the following sections of this chapter are based on a scenario in which the payer and the payee represent one pair of participants along a cross-border payment chain. Both the payer and the payee have accounts on at least one common ledger, on which they agree to use the payment method.

4.1 Trustline

Trustline is a payment method that is entirely based on trust between the payer and the payee. Individual trustline payments are not settled on the ledger, with only the final settlement occurring at a later stage. The fund transfer process using trustline can be broken down into three phases: setup phase, state update phase, and settlement phase.

Setup phase: A participant (e.g. Entity B) can open a trustline with another participant (e.g. Entity A) that has an account on the same ledger, by setting a maximum amount that can be sent from A to B without settlement.[34]

State update phase: The payer prepares the payment by sending the payee a message including the hash value and timeout. The total amount of unsettled payments, or the *state* of the trustline, is kept by both participants in their respective databases. Technically, a trustline could be used for payments in both directions, provided that there is enough available credit or the maximum trusted amount is not exceeded. If Entity A prepares a payment with a hash value and Entity B provides the hash preimage to Entity A before the timeout, Entity B's balance within the state of the trustline is increased by the payment amount (and Entity A's balance is decreased). On the other hand, if Entity B prepares a payment with a hash value and Entity A provides the hash preimage to Entity B before the timeout, Entity A's balance is increased by the payment amount (and Entity B's balance is decreased). Payments are not settled in this phase.

Settlement phase: The total amount of unsettled payments is settled by making a transfer on the ledger.

[33] The current ILPv4 specification distinguishes between main HTLA types and additional HTLA types (third party escrow, notarised payment channel and third party payment channel). Additional HTLA types rely on a third party that is trusted by participants involved in the payment method. As these methods do not seem to need additional requirements on the part of the ledger, we take up third party escrow as the typical example of the three.

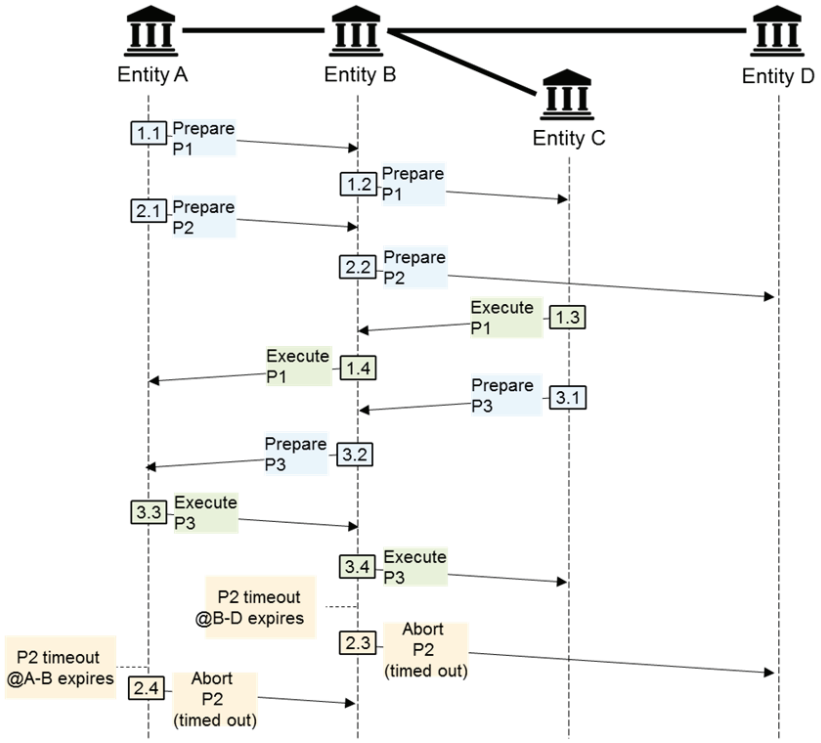
[34] Alternatively, a trustline can be set up in a prefunded way that would first require Entity A to transfer the maximum trusted amount on the ledger to Entity B. How a trustline is set up may matter in scenarios when one party trusts the other, but not the other way around.

Participants can use a trustline on any type of ledger, because the only requirement towards the ledger is the ability to make transfers.

A trustline is an off-ledger payment method, i.e. it does not require each single payment to be settled on the underlying ledger. For this reason, both entities may continue to make payments without settlement provided there is enough available credit or the maximum amount accepted as debt is not exceeded. Payments can reconvene once the outstanding debt is settled. Using the trustline enables the payer and the payee to transact independently from the ledger availability, ledger throughput, or latency.

In order to clarify the functioning of the trustline and other off-ledger payment methods, a scenario with multiple concurrent payments (some executed and others aborted) among four network participants (A, B, C and D) is introduced in Figure 10. There are multiple payment chains, but trustline is used in every payment between A and B. The details of the payments are described in the following figure and table.

Figure 10
Example of concurrent payments



Payment chain	Sender	Receiver	Amount	Preimage/value
P1	A	C	50	S1 / H1
P2	A	D	10	S2 / H2
P3	C	A	12	S3 / H3

The maximum amount that B trusts A with is equal to €100.00. With netting agreement between entities A and B, the state of the trustline between A and B is updated according to the following table.

Table 2
Updates of state of trustline accounting for concurrent payments

Event	Description	State of the A - B trustline			
		Funds available for A to prepare	Funds available for B to prepare	Prepared A → B	Prepared B → A
	Initial state	€100.00	€0.00	€0.00	€0.00
1.1	P1 is prepared	€50.00	€0.00	€50.00	€0.00
2.1	P2 is prepared	€40.00	€0.00	€60.00	€0.00
1.4	P1 is executed	€40.00	€50.00	€10.00	€0.00
3.2	P3 is prepared	€40.00	€38.00	€10.00	€12.00
3.3	P3 is executed	€52.00	€38.00	€10.00	€0.00
2.4	P2 is aborted	€62.00	€38.00	€0.00	€0.00

4.2 On-ledger escrow using HTLC

On-ledger escrow using HTLC allows conditional transfer that is enforced by the ledger. Payer's funds are put on hold by the ledger, pending the fulfilment of a predefined condition. The HTLC stipulates that the payee may claim the funds by presenting a valid preimage of the hash value before the given timeout. If the preimage is not presented to the ledger before the expiration of the HTLC timeout, then the funds are returned to the payer.[35]

The conditional transfer with HTLC is initiated by putting funds on hold for a certain period of time, using a value of a cryptographic hash function. Both the timeout and the value are chosen by the payer on a per-payment basis. Within this period of time funds can only be released to the payee upon the fulfilment of a given condition, namely the provision of a valid preimage of the hash value that was used to put the funds on hold. If, however, the predefined time period passes without the fulfilment of the condition, the funds are returned to the payer.[36]

On-ledger escrow can only be implemented on ledgers that support HTLC functionalities, i.e. the ledger must be able to process HTLC. For this reason, this payment method can only be used on ledgers that provide this specific capability. See Annex 1 for more details about ledger requirements. It should also be noted that each single payment is settled on the ledger, implying that on-ledger escrow can only be used when the ledger is available and the latency and volume of transactions are affected to a large extent by the ledger throughput and latency.

[35] Depending on the implementation, the payer may need to claim the funds after the timeout expires, by sending a refund instruction to the ledger.

[36] In order to shorten the time period in which funds are locked by the ledger in cases where a prepared transfer with on-ledger escrow is rejected by the payee, participants can have an additional agreement that releases the funds to the payer upon the payee's request (payment reject scenario).

4.3 Third party escrow

The third party escrow is a payment method that relies on a trusted third party in order to create a transfer that is conceptually similar to the on-ledger escrow using HTLC.

The payer sends the information for the prepared payment to a third party and transfers funds to an account owned by the third party that is distinct from the ledger operator, but trusted by both the payer and the payee. The third party transfers the escrowed funds to the payee once the payee presents the valid preimage to the hash value within the specified period of time. If the preimage is not presented before the predefined time period passes, funds are returned to the payer.[37]

For the use of a third party escrow, only the ability to make transfers is required from the ledger. This allows participants to use a third party escrow with any type of ledger. Third party escrow is also an on-ledger payment method. This means that it can be used only when the ledger and the third party are available and the latency and volume of transactions are affected by the ledger and the third party throughput and latency.

4.4 Payment channels

A payment channel allows two parties to combine multiple payments between each other and only settle the final net position. The payment channel represents a relationship between two parties that is supported by the ledger in which they have accounts. Payment channel is an off-ledger payment method. Therefore, fund transfers using payment channel could be also broken down into three phases: setup phase, state update phase, and settlement phase.

Setup phase: The relationship is established by opening a payment channel when one or both parties involved in the payment put a certain amount of funds on hold, usually by escrowing it in a shared, temporary account.

State update phase: The individual payments over the open payment channel are effectively made by sending signed claims that entitle each party to a specified portion of funds held in the shared account. The entitlements are called payment channel *state.* Signed claims are both verifiable by participants or the ledger and enforceable by the ledger. Each new claim represents an updated version of the entitlements and makes all previously exchanged claims outdated. Parties send claims directly to each other without making any funds transfer into or out of the shared account on the ledger. It should be noted that payment channels could be

[37] Note for third party escrow, in order to shorten the time period in which funds are locked by the third party in the case that a prepared payment is rejected by the payee, participants can have an additional agreement that releases the funds to the payer upon the payee's request.

used for payments in both directions, provided that there is enough capacity in the payment direction.[38]

Settlement phase: The payment channel is settled and the relationship is terminated when the shared account is closed and funds held in that account are distributed between both parties as specified in the most recent claim that contains the final net position. The payment channel can be closed both unilaterally (non-cooperatively, usually in the case of disputes between participants) and bilaterally (cooperatively).

Only the opening and closing of payment channels is recorded on the underlying ledger in form of transfers into and out of the shared account, whereas the exchange of claims is done bilaterally, and off-ledger. This allows successful payments to flow independently from the ledger availability, ledger throughput or latency, while in cases of disputes the involvement of the ledger may be required.

Payment channels can only be implemented on ledgers that support payment channel functionalities and enforce the settlement of the final net position, on which participants have agreed (i.e. the latest payment channel state).

The use of payment channels entails the exchange of a signed claim for a certain secret. In the following subsections we describe two types of payment channels that can be used as payment methods: simple payment channel and conditional payment channel.

4.4.1 Simple payment channel

In the case of a simple payment channel, the payer promises to deliver a signed claim to the payee if a valid preimage of a predefined hash value is presented within a predefined period of time. The received claim will entitle the payee to a larger portion of funds held in the shared account. The delivery of the promised claim in exchange for a hash preimage, the verification of the validity of that preimage and that of its timely submission are not enforced by the ledger but are instead at the discretion of the payer, and therefore based on trust that the payer will keep its promise.

The main difference between the trustline and the simple payment channel is that, at any point in time, each participant can choose to submit the latest signed claim it has to the ledger, which will enforce the settlement of the latest payment channel state that was agreed.

From a technical perspective, in addition to payment channel functionalities, there are no additional requirements for this specific payment method.

[38] In the report, bidirectional and symmetric payment channels are considered. Bidirectional means that payments can flow in both directions, while symmetric means that a payment channel claim is valid for both participants of the payment channel and participants store exactly the same information about payment channel states. Some existing implementations are unidirectional or asymmetric and have other requirements and process flows than the bidirectional and symmetric ones.

In order to clarify the functioning of the simple payment channel, with reference to the payments example in Section 4.1, we assume that the payment method between A and B is a simple payment channel, initially funded by A, with a capacity of €100.00. The state of the payment channel changes according to the following table.

Table 3
Updates of state of simple payment channel accounting for concurrent payments

Event	Description	State of the A – B simple payment channel[39]		Not covered by the simple payment channel claim	
		A balance in the claim	B balance in the claim	Prepared A → B	Prepared B → A
	Initial state	€100.00	€0.00	€0.00	€0.00
1.1	P1 is prepared	€100.00	€0.00	€50.00	€0.00
2.1	P2 is prepared	€100.00	€0.00	€60.00	€0.00
1.4	P1 is executed	€50.00	€50.00	€10.00	€0.00
3.2	P3 is prepared	€50.00	€50.00	€10.00	€12.00
3.3	P3 is executed	€62.00	€38.00	€10.00	€0.00
2.4	P2 is aborted	€62.00	€38.00	€0.00	€0.00

4.4.2 Conditional payment channel with HTLC

While a conditional payment channel is similar to a simple payment channel in that it enables the payer and the payee to exchange off-ledger signed claims, the payment channel claim in a conditional payment channel includes HTLC, which is enforceable by the ledger. This removes the need for trust between the two parties because the fulfilment of spending conditions for the conditional payment is not at the discretion of one party.

The main difference between the simple payment channel and the conditional payment channel is that HTLC are part of the payment channel state that is agreed between the parties via the claims. This means that at any time when the ledger is available, each of the participants can send the latest claim to the ledger and transform the off-ledger HTLC (in the claim) into an on-ledger HTLC that is conceptually identical with the one described in Section 4.2.

From a technical perspective, in addition to payment channel functionalities, the underlying ledger would be required to have HTLC capabilities.

In order to clarify the functioning of the conditional payment channel, with reference to the payment example in Section 4.1, we assume that the payment method between A and B is a conditional payment channel, initially funded by A, with a

[39] During the preparation phase, prepared funds could be assigned to one of the counterparties (payer, payee, Entity A regardless of the payer, Entity B regardless of the payer) or a third party, depending on trust relationships between Entities A and B. Within this report we consider a specific simple payment channels design where the payee trusts the payer and prepared funds are always assigned to the payer's balance in the claim, leading to unchanged balances during payment preparations.

capacity of €100.00. The state of the payment channel changes according to the following table.

Table 4
Updates of state of conditional payment channel accounting for concurrent payments

Event	Description	State of the A - B conditional payment channel			
		A balance in the claim	B balance in the claim	Total HTLC A → B in the claim	Total HTLC B → A in the claim
	Initial state	€100.00	€0.00	€0.00	€0.00
1.1	P1 is prepared	€50.00	€0.00	€50.00	€0.00
2.1	P2 is prepared	€40.00	€0.00	€60.00	€0.00
1.4	P1 is executed	€40.00	€50.00	€10.00	€0.00
3.2	P3 is prepared	€40.00	€38.00	€10.00	€12.00
3.3	P3 is executed	€52.00	€38.00	€10.00	€0.00
2.4	P2 is aborted	€62.00	€38.00	€0.00	€0.00

4.5 Ledger requirements for payment methods

All ledgers need basic functionalities such as user authentication, and sending and receiving funds between accounts, in order to process transfers. In addition, some of the payment methods described in previous sections require underlying ledgers to have specific functionalities, and are therefore only available to participants that have their accounts on such ledgers. These functionalities are those associated with processing HTLC and processing payment channels.

- The ability to process HTLC involves locking funds for a given amount of time and releasing them according to the conditions defined in the HTLC.
- The ability to handle payment channels is to lock funds until a participant (or both of them) decides to redistribute them as specified in the latest claim that was agreed off-ledger.

For more details on the last two functionalities see Annex 1.

4.6 Summary

In this chapter, we provided an overview of the payment methods taken up in this study. These five methods show distinctive characteristics which can be summarised as follows: (i) whether individual payments are settled on-ledger or recorded off-ledger, (ii) whether funds are locked or escrowed, (iii) whether payments are enforced when the predefined condition for the payment is fulfilled, and (iv) whether specific ledger functionalities are required to conduct transfers, such as the functionalities to enforce conditional transfers and process signed claims. This could be summarised as in the table below.

Table 5
Overview of payment methods and specific ledger requirements

Payment method	On-ledger/ Off-ledger	Escrow/ Lock	Enforcement of conditional payment	Specific ledger requirements	
				To process Hashed Timelock Contracts	To process payment channels
Trustline	Off-ledger	No	No enforcement	No	
On-ledger escrow	On-ledger	Yes	Enforced by ledger	Yes	No
Third party escrow	On-ledger	Yes	Enforced by third party	No	
Simple payment channel	Off-ledger	Yes	No enforcement	No	Yes
Conditional payment channel	Off-ledger	Yes	Enforced by ledger[40]	Yes	Yes

5 Experimentation[41]

5.1 Overview of experiments

In Stella phase 3, we conducted experiments to examine the feasibility of synchronised settlement across different ledgers using the ideas introduced in the Interledger whitepaper about universal interledger payments. Experiments were conducted with and without the ILP, one of the most relevant specifications based on the whitepaper. The experiment without ILP was carried out for funds transfers between DLT ledgers, while experiments with ILP was for fund transfers between DLT ledgers, between centralised ledgers, and between a DLT ledger and a centralised ledger. The knowledge obtained from the experiments complements the main results of the conceptual study conducted in this report. Research on performance was not within the scope of these experiments.

We identified two publicly available open-source implementations of ILP: Interledger.js[42] and Hyperledger Quilt.[43] Interledger.js is a reference JavaScript implementation of the ILP stack with an active community of developers that work on different solutions with different blockchains. Hyperledger Quilt is a Java implementation of ILP and is currently in incubation under the umbrella of the

[40] Enforcement by the ledger occurs only in the case when the payment channel is closed.
[41] See Annex 2 for the details of experiments.
[42] https://github.com/interledgerjs/
[43] https://github.com/hyperledger/quilt/

Hyperledger Foundation. Interledger.js was adopted in this phase, mainly because of the availability of documentation.

The specification of ILP is currently developed by the Interledger Payment Community Group of the World Wide Web Consortium (W3C) after the release of the Interledger whitepaper.[44] The latest version of the protocol as of May 2019 is ILPv4. Nevertheless, for our experimentation, an older and deprecated version of ILP was adopted, namely ILPv3. This choice was made mainly due to unavailability of an open-source centralised ledger implementation of ILPv4 at the time of the initial experimentation.

In particular, as an example of a centralised ledger, the Five Bells Ledger was used as implementation of ILPv3 ledgers. Its source code is openly available as a part of Interledger.js, together with its ILP plugin (a piece of software that is used by clients to communicate with the ledger). As for a DLT ledger, Hyperledger Fabric was adopted similarly as in the previous phases of Project Stella. However, its ILP plugin was newly implemented during the experiments. Since Five Bells Ledger supports on-ledger escrow, we also implemented it on the Hyperledger Fabric ledgers.

5.2 Experiment with centralised ledgers

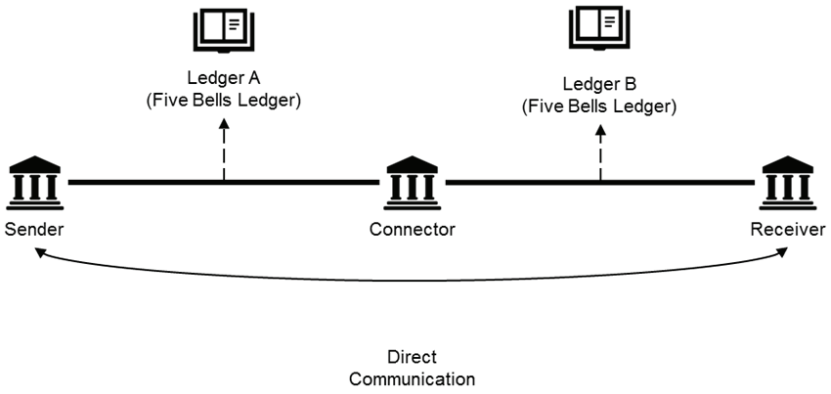
We conducted experiments to confirm synchronised settlement between centralised ledgers using ILP. Five Bells Ledger[45] was adopted as a centralised ledger and ilp-plugin-bells[46] was used as a plugin for client applications of participants to connect to Five Bells Ledger. This experiment confirmed that, in the absence of failures, synchronised settlement by on-ledger escrow can be conducted between centralised ledgers using ILP.

[44] https://interledger.org/

[45] https://github.com/interledger-deprecated/five-bells-ledger

[46] https://github.com/interledger-deprecated/ilp-plugin-bells

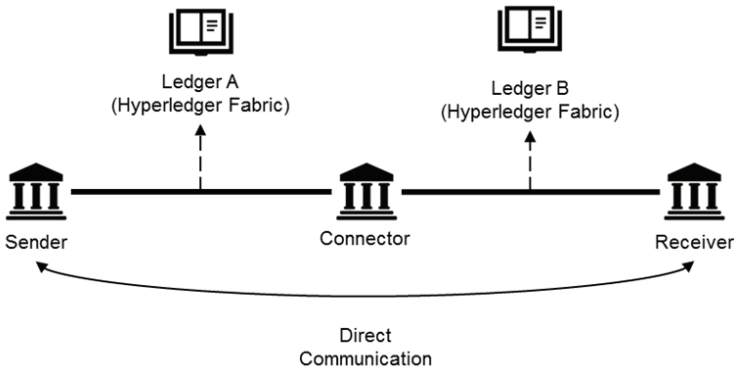
Figure 11
Experiment setup on centralised ledgers



5.3 Experiments with DLT ledgers

Hyperledger Fabric was used in the experiments on DLT ledgers as in the previous phases of Project Stella. Hyperledger Fabric version 1.2.1 was used in Stella phase 3, an update from 1.1.0-alpha previously utilised in Stella phase 2.[47]

Figure 12
Experiment setup on DLT ledgers



5.3.1 Experiment without ILP

Funds were transferred from the Sender, who only has an account on Ledger A, to the Receiver, who only has an account on Ledger B through the Connector, who has

[47] Hyperledger Fabric Client SDK for NodeJS was used in this study.

accounts on both Ledgers A and B. In the absence of failures, synchronised settlement between different ledgers was achieved without ILP as expected. This was done by locking funds to be transferred from the Sender to the Connector and from the Connector to the Receiver by using on-ledger escrow with the same hash value and by releasing them with the same preimages. In this experiment, the Connector does not implement any client applications for the foreign exchange of funds transferred although it could be implemented by modifying the ratio of the amounts received and sent.

5.3.2 Experiment with ILP

In order to carry out experiments on DLT ledgers, we implemented some ledger functionalities[48] within Hyperledger Fabric, together with a plugin that allows participants to connect to the ledger using ILP functionalities. The experiment confirmed that synchronised settlement, which consists of two on-ledger escrow using HTLC on different ledgers, can be achieved.

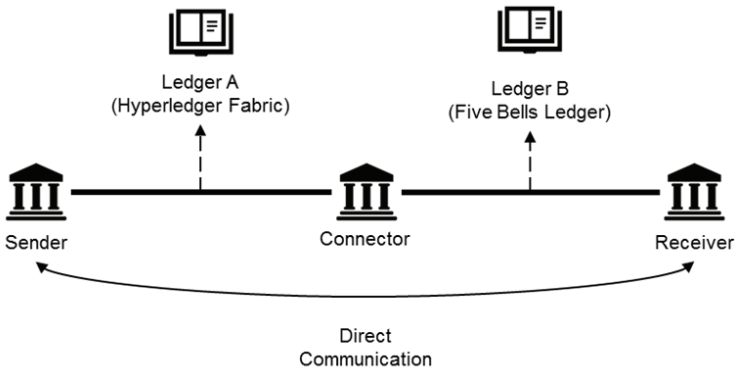
5.4 Cross-platform experiment

We conducted an interledger payment in a payment chain that involves a centralised ledger using Five Bells Ledger and a DLT ledger using Hyperledger Fabric to confirm that ILP does not depend on the specific technology of the ledger. Specifically, funds were transferred from the Sender who only has an account on Hyperledger Fabric ledger (Ledger A) through the Connector who has an account on Hyperledger Fabric as well as on Five Bells Ledger (Ledger B) to the Receiver who only has an account on Five Bells Ledger.

In the figure below, Ledger A is the same ledger as Ledger A in Section 5.3.2, while Ledger B is the same ledger as the Ledger B in Section 5.2. We added a plugin to the client application of the Connector of Section 5.3.2. Due to the standardisation of the ILP interface, modification of the ledger and plugin was not necessary.

[48] Hash function has been made compatible to ILP. SHA256 is used in this study.

Figure 13
Cross-platform experiment setup with ILP



5.5 Results of experiments

Several patterns of synchronised settlement have been tested with successful results, including settlement between DLT ledgers, between centralised ledgers and between a DLT ledger and a centralised ledger using ILP, a specification of the transfer protocol introduced in the Interledger whitepaper. Synchronised settlement without using ILP on DLT ledgers is also feasible, and therefore using ILP is not a necessary requirement for synchronised settlement. Nevertheless, ILP could facilitate the abstraction of different types of ledgers, and therefore could bring about the benefit of standardisation.

6 Assessment of payment methods

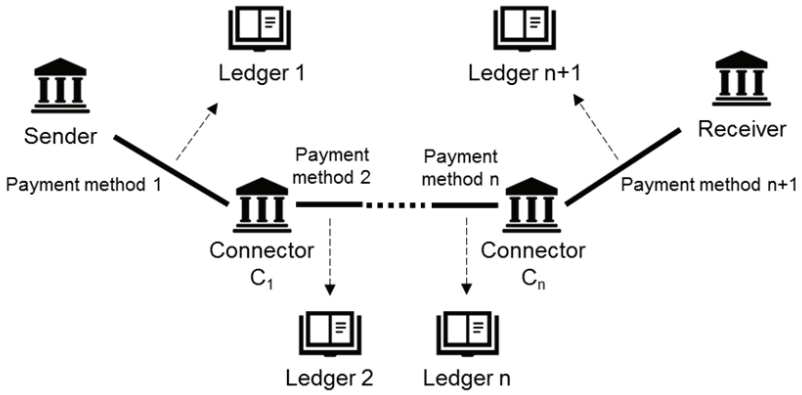
This chapter assesses the safety and efficiency of individual payment methods. While the payment methods introduced in Chapter 4 synchronise payments, this characteristic does not necessarily ensure that the payments are safe from credit risk – a payment method can be considered safe if it guarantees that the participant (Sender, Receiver, or Connector(s)) who completely satisfies its responsibilities in the transaction process is not exposed to the credit risk of incurring a loss on the principal amount being transferred. We assess the safety of each payment method using a failure scenario. Then, we perform a basic analysis on liquidity efficiency.

6.1 Safety

In this section, we consider a payment case where the Sender conducts a payment to the Receiver which routes through Connectors C_1, C_2, …, and C_n (C_1 receives the funds from the Sender and C_n sends the funds to the Receiver). The Sender,

Connectors and the Receiver follow the protocol, the concept of which is explained in Chapter 3.

Figure 14
Payment chain



In the protocol for interledger payments, the payment from C_{i-1} to C_i is supposed to occur after the payment from C_i to C_{i+1}. In this case, the default of C_{i-1} after the payment from C_i to C_{i+1} could pose a risk to C_i.[49] Therefore, we conduct analysis on the safety implications of the following failure scenario for payments conducted by each of the payment methods: C_{i-1} becomes insolvent and unable to pay after C_i has already made payment to entity C_{i+1} but before C_i receives the funds from C_{i-1}.[50]

When conducting this assessment, we assume that the following preconditions are met. Failure to meet these preconditions means payments cannot be safely conducted regardless of the payment method considered.

I. All participants of the payment chain follow economic incentives.

II. The funds deposited in the ledgers on which the participants hold their accounts must be secure. For example, if the ledger defaults during the payment process, the funds may be lost, regardless of the payment method.[51]

III. The ledgers (on which the participants hold their accounts) could be trusted to fulfil their responsibilities for the successful payment.[52]

[49] This is in contrast to most current cross-border payments not following the protocol, in which the payment from C_{i-1} to C_i occurs before the payment from C_i to C_{i+1}. In this case, the default of C_{i-1} does not pose a risk to C_i but may pose risks to C_j ($j \leqq i-2$, C_j could be the Sender), depending on legal arrangements, etc.

[50] Note that C_{i-1} could be the Sender and C_{i+1} could be the Receiver.

[51] If third party escrow is used, whether safe payments could be made in the case of the default of the provider of third party escrow depends on if the escrow process creates the liability of the provider to the payer/payee. In the third party escrow method discussed in this report, the payer sends funds to the account held by the provider of third party escrow for them to be escrowed, so the payment is vulnerable to the default of the provider. On the other hand, it may be possible to consider a third party escrow procedure where the payer sends funds to the joint account in which signatures of the payer and the payee are required to transfer and the third party as an agent of the payer and the payee instructing the ledger to hold and release the funds, without creating its liability to the payer/payee.

IV. The ledgers which put funds on hold have sufficient processing speed and operation hours.

V. The participants and ledgers involved in the transaction possess the required capabilities outlined in Chapter 4[53] and can fulfil their responsibilities for the transfer (i.e. conduct the payment procedure outlined in Sections 4.1 to 4.4, even in severe stress scenarios).[54]

VI. The payee (the Receiver and Connector(s)) should have sufficient security and other capabilities in place to ensure that the preimage and other important information could be disclosed when and only when it is required in the payment process.

VII. When funds are put on hold, the funds are isolated from the default of the payer.

VIII. We also assume that providers of third party escrows meet the preconditions above which apply to ledgers.

Based on this scenario, C_i's ability to receive the relevant funds from C_{i-1} depends on the payment method being used. That is, C_i would be able to receive the funds if on-ledger escrow, conditional payment channel, or third party escrow is used, while it might not be able to receive the funds if trustline or simple payment channel is used.

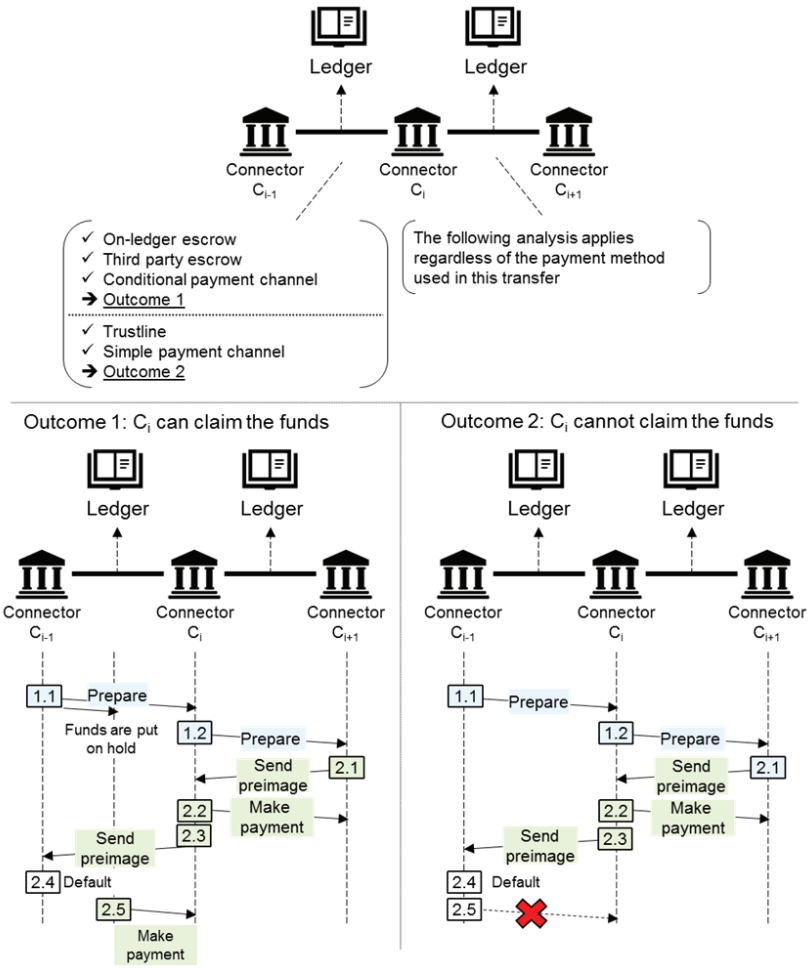
This is due to the fact that in the former group of payment methods, the funds are put on hold in a secure manner before C_{i-1} becomes insolvent and there is an enforcement mechanism which ensures that appropriate transfers occur according to whether the payee fulfils the predefined condition or not, whereas in the latter group the payment is based purely on a promise by C_{i-1}. The following figure illustrates this:

[52] If this is not the case, the participants would not have accounts on these ledgers in the first place.

[53] This includes having a synchronised timekeeping mechanism that participants can rely on.

[54] Force majeure scenario – where the payee is unable to send the preimage before the timeout – should be understood as the payee failing to satisfy its responsibilities as stipulated in precondition V, and therefore is out of scope for this assessment.

Figure 15
Scenario involving a defaulted counterparty



Note that for conditional payment channel (Outcome 1), the funds used for the payment have already been put on hold by the ledger when the payment channel was created.

While both trustline and simple payment channel cannot ensure safe transfer under this failure scenario, there could be a difference between the two methods with regard to the amount of loss C_i may incur. While C_i may lose only the amount that is not reflected in the most recent claim in the case of simple payment channel, it could lose all credit to C_{i-1} under a trustline arrangement.[55]

[55] Additional considerations are required for the case where Connector C_n, which is supposed to send the funds to the Receiver, becomes insolvent after it has already received the preimage of the hash generated by the Receiver, but before the Receiver receives the funds from C_n.

6.2 Liquidity efficiency

Our second part of the assessment looks at the liquidity efficiency of each payment method.[56] The five payment methods considered in this report can be categorised into three groups: (1) trustline, (2) on-ledger escrow and third party escrow, and (3) simple and conditional payment channels – in the order of liquidity efficiency.

Trustline is the only post-funded model out of the five payment methods. Since the payer using trustline does not need to prepare funds ex ante and could net payments with the payee, this payment method is the most liquidity-efficient.

Among the other four pre-funded models, we find that the liquidity efficiency of on-ledger escrow and third party escrow is generally superior (in most cases strictly better) than that of simple and conditional payment channels. While the former methods require the payer to put on hold the transfer amount for each transfer, the latter require the payer to put on hold an amount for transfers for the entire duration of a payment channel that is greater than (or equal to)[57] the amount for a single transfer.[58]

This could be demonstrated through the case where the payer plans to pay 1 unit of money at 1 p.m. and 3 units of money at 2 p.m. If the payer uses a payment channel for these payments, the payer needs to prepare all 4 units of money before 1 p.m. If the payer uses on-ledger escrow or third party escrow for these payments, the payer only needs to prepare 1 unit of money before 1 p.m. and could prepare the additional 3 units of money before 2 p.m.

6.3 Summary

Through the safety analysis in Section 6.1, we have highlighted safe payment methods which can ensure that participants involved in the payment (the Sender, Receiver, or Connector(s)) who completely satisfy their responsibilities in the transaction process are not exposed to the risk of incurring a loss on the principal amount being transferred.[59] This group of payment methods which ensures safe cross-ledger payments include on-ledger escrow, conditional payment channel, and

[56] Liquidity management considerations, such as methods to fund accounts and balancing funds held on multiple ledgers, are not considered. This is because while these considerations may have implications on liquidity efficiency, they are unrelated to the features of each payment method and therefore do not differentiate one payment method from another.

[57] If payment channel arrangement is made for a single transfer, then the difference in liquidity efficiency between a payment channel and on-ledger escrow or third party escrow diminishes.

[58] Note that this assessment is based on the assumption that the ledgers which put the funds on hold have sufficient processing speed and operating hours. Also, while it may be technically possible for the participants to increase the amount put on hold in the payment channel based on mutual agreement, this case is also out of the scope of this report as it would undermine the benefit of payment channel (updating balances off-ledger).

[59] It should be noted that the Sender does not risk losing the principal amount of the transfer regardless of the payment method used, since when all participants in cross-ledger payments using a protocol proposed in the Interledger whitepaper satisfy their responsibilities, the Sender pays the funds only when the Receiver has already received the payment. On the other hand, as explored in the failure scenarios, the Receiver and Connector are exposed to the risk of losing the principal amount of the transfer, depending on the payment method being used.

third party escrow (Default-resistant Conditional Transfers (DCT)). While participants involved in the payment may choose to use trustline and simple payment channel at their own risk, they should be aware of their exposure to the default risk of the payer.

As for liquidity efficiency, our analysis in Section 6.2 has shown that the liquidity efficiency of trustline is the most efficient among the five payment methods while the liquidity efficiency of on-ledger escrow and third party escrow is better than that of simple payment channel and conditional payment channel.

The results of the analysis above can be summarised as in the following table.

Table 6

Overview of assessment results

Payment method	Safety	Liquidity efficiency
On-ledger escrow using HTLC	Good	Fair
Third party escrow	Good	Fair
Conditional payment channel with HTLC	Good	Poor
Simple payment channel	Poor	Poor
Trustline	Very Poor	Good

Note: As for safety, "Good" is assigned to DCTs, "Poor" indicates that participants could lose the amount that is not reflected in the most recent claim, and "Very Poor" indicates that they bear the risk of losing all credit to the counterparty.
As for liquidity efficiency, "Good" is assigned to post-funded methods, "Fair" is assigned to pre-funded methods which require amounts of individual payments to be prepared, and "Poor" is assigned to pre-funded methods which require an amount equal to or greater than that of the individual payment.

7 Additional considerations

In this chapter, we discuss three considerations we deem relevant relating to the protocol for interledger payments. Namely, we address (i) the distinction between safety of individual payments and atomicity of the payment chain, which are related but distinct concepts that require clarification, (ii) a consideration on limitations on ledger processing speed and operating hours, which were not addressed in the assessment, and (iii) the “free option problem”, which is an issue that could occur even when all of the preconditions used in the assessment are met, and without the safety of payments being undermined.

7.1 Safety of individual payments and atomicity of the payment chain

In Chapter 6, we analysed the safety of individual payment methods for participants. Safety of individual transfers should not be confused with the atomicity of all transfers within a payment chain. All transfers within a payment chain could be considered atomic if each transfer is executed if and only if all other transfers of the payment chain are executed.[60]

[60] We do not assess the atomicity of all payments of a payment chain in detail because the atomicity of all payments within a payment chain does not ensure the safety of all transfers within a payment chain.

Atomicity of all transfers within a payment chain could only be realised if all of the individual transfers along the payment chain are safe. To illustrate this, recall the failure scenario in Section 6.1. Under the preconditions highlighted in Chapter 6, if all individual transfers along the chain are conducted using safe payment methods, then all transfers would occur even in a failure scenario with a defaulted counterparty, as the participants (including the defaulted party, or its creditor) would execute the transfer once it is initiated by the Receiver. If the Receiver does not initiate the transfer, then all funds transfers along the payment chain will be aborted due to timeout.

If, instead, unsafe methods are used in the payment chain, under the same failure scenario, the atomicity of all transfers within the payment chain may not be achieved. But even in this case, transfers between participants who used safe payment methods could still be executed and their funds would be safe.

Finally, it should be noted that the safety assessment in Chapter 6 was conducted based on several preconditions. According to some of them, all participants of the payment chain follow their economic incentives (precondition I), and all ledgers have sufficient processing speed and availability (precondition IV). It should be noted that, in the absence of these preconditions, the atomicity of the payment chain cannot be guaranteed regardless of payment methods used along the chain, as individual transfers may no longer be considered safe.

7.2 Impact of ledger processing speed and operational availability

In Chapter 6, we assumed that the ledgers have sufficient processing speed and operational availability (precondition IV). However, in general, participants need to take ledger processing speed and operating hours into account when deciding on the timeouts. Concretely, if an entity uses a payment method where the ledger is involved (on-ledger escrow, third party escrow, and conditional payment channel), then the entity needs to take ledger availability and processing time into account when it sets the timeout.

Usage of a payment method that involves a ledger which is slow or unavailable for most of the day in one payment may affect timeout considerations for other payments along the payment chain. For example, assume that the payment between C_i and C_{i+1}(C_{i+1} could be the Receiver)) is conducted using a ledger that is very slow, and that C_j ($j<i$, C_j could be the Sender) somehow expects the payment to route through C_i and C_{i+1}. Then, C_j may decide to set the timeout between itself and C_{j+1} in a way that provides enough time for C_i and C_{i+1} to execute the payment on the slow ledger (otherwise, C_i may not accept routing the payment altogether, fearing that it would not be able to submit the preimage to C_{i-1}).

In order to shorten payment timeouts, it is necessary to either use payment methods that do not involve ledgers in the payment process (i.e. trustline or simple payment

channel)[61], or improve the processing speed and/or availability of ledgers. However, it should be noted that the users of trustline and those of simple payment channel could face an increased risk as the safety of their funds would be undermined.

7.3 The free option problem

The "free option problem" refers to the exchange rate risk that participants are exposed to when relaying a cross-ledger, cross-currency payment. In the preparation of such a payment, participants enter into a commitment to deliver a fixed amount denominated in one currency in exchange for a fixed amount denominated in another currency. This could potentially be exploited by malicious actors.

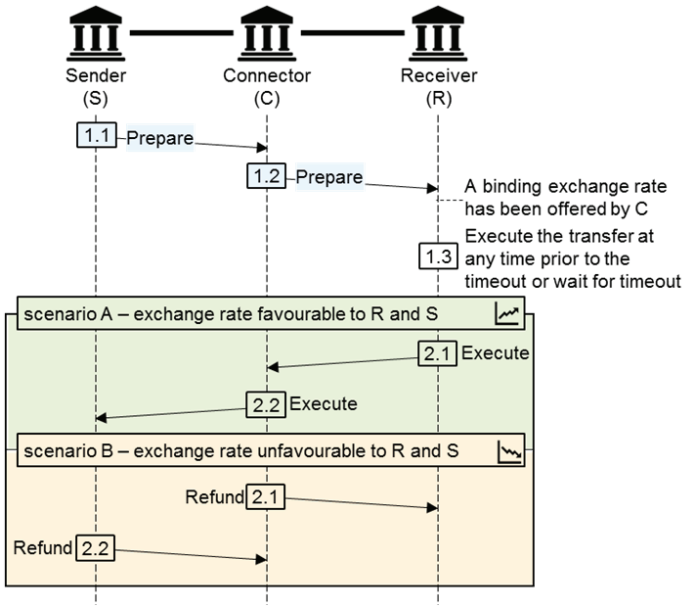
The figure below shows an example in which the Sender and the Receiver collude after initiating a transaction and thereby effectively locking the Connector's liquidity. In this instance, the Receiver has the option to either execute the payment (fulfil), or alternatively abort the payment by rejecting it within the timeout or by letting the timeout expire. Thus, the colluding parties can take advantage of the Connector's binding commitment depending on the movement of the exchange rate. That is, they only execute the payment if the exchange rate moves in a favourable direction for them, otherwise the payment is aborted.

This free option problem currently remains open and is being actively discussed within the Interledger community. It should be noted that this issue arises due to the design of the protocol and would occur even when all of the preconditions in Chapter 6 have been met. However, it should also be noted that this issue does not pose direct risk to the safety of the payments. Moreover, this may not be a significant issue if the participants value reputational risk above the potential gains from exploiting the Connector's binding commitment.

[61] In this case, procedures for off-ledger payment methods must be conducted with sufficient speed.

Figure 16

Example scenario: exploitation of a “free option”



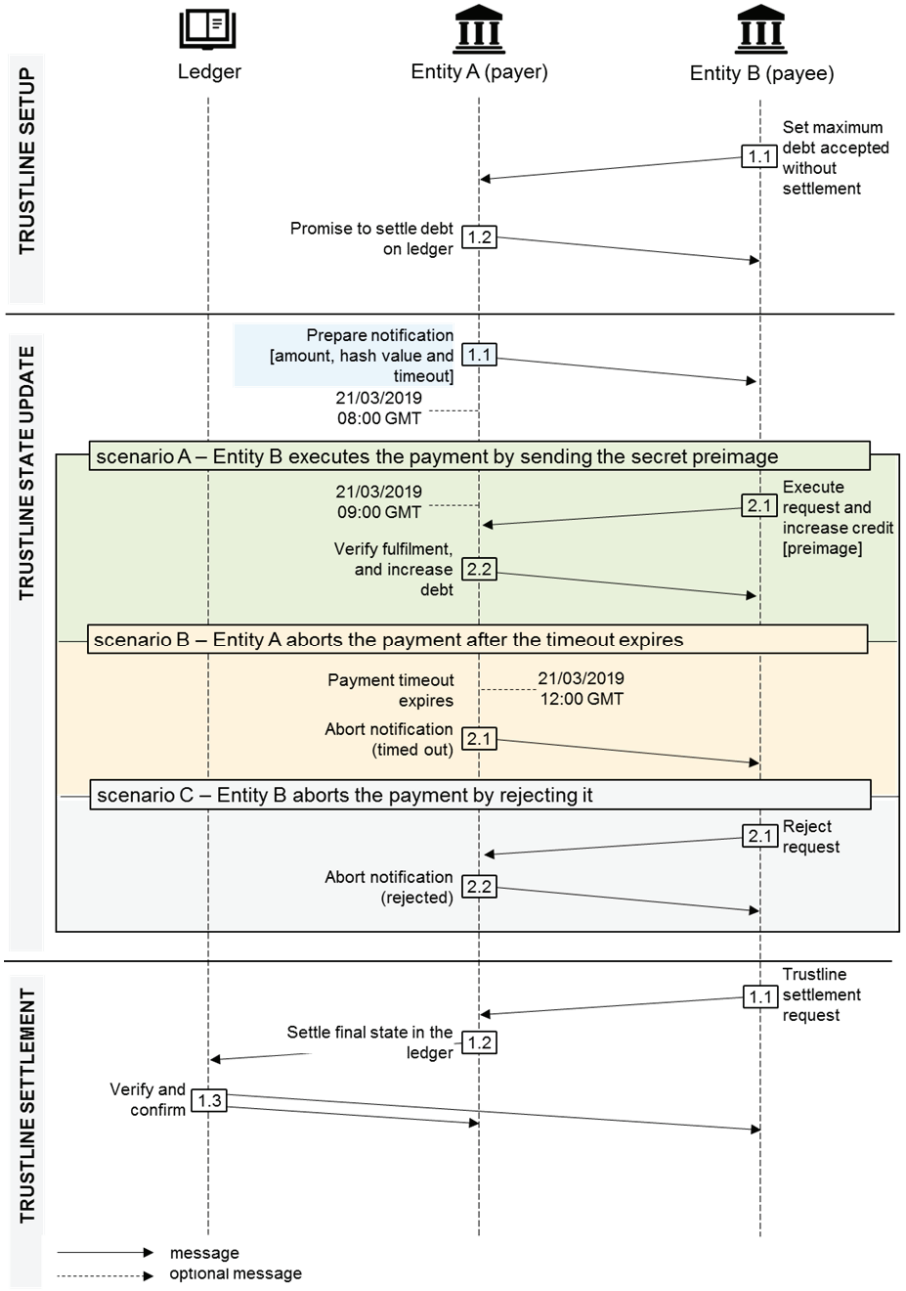
Annex 1: Process flows and ledger requirements

This annex contains a potential functioning of the payment methods, detailed with figures. It also contains a description of high level requirements for the ledger functionalities that were identified in Chapter 4 of the main text. Figures, process flows and requirements should not be understood as a specification but as a tool for the clarification of the payment methods that were introduced in the main text.

Trustline

Trustline setup in Figure A shows Entity B opening a trustline with Entity A. Then the state of the trustline is updated off-ledger. Finally, Entity B requests the settlement of the trustline state on the ledger.

Figure A
Trustline process flow

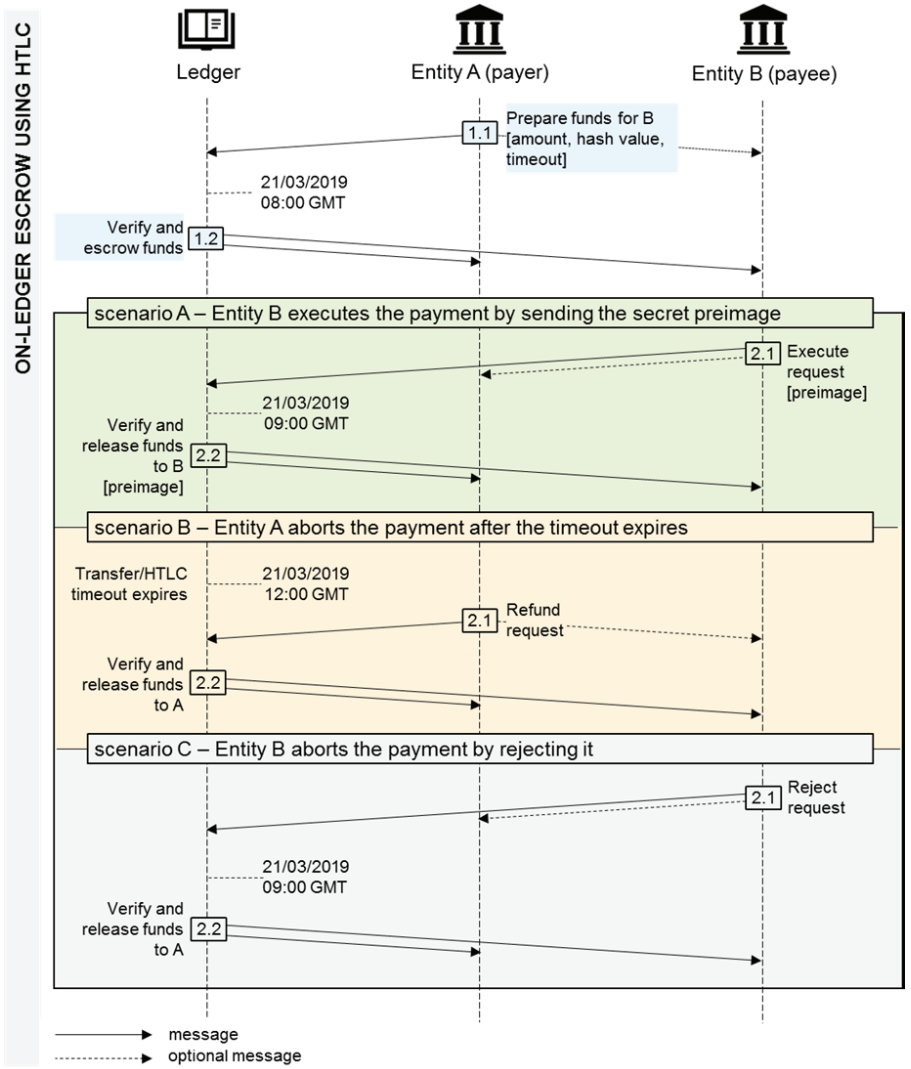


On-ledger escrow using HTLC

A single payment using on-ledger escrow using HTLC is illustrated in Figure B. The preparation request is sent to the ledger and optionally also to Entity B (depending on the arrangement). Execute, Refund and Reject request are also sent directly to the ledger by the participants, and optionally to the counterparty. In general, bilateral communication between participants is not required for on-ledger escrow using HTLC but could be desirable for participants to exchange additional information and not only the amount, the hash value and preimage.

Figure B

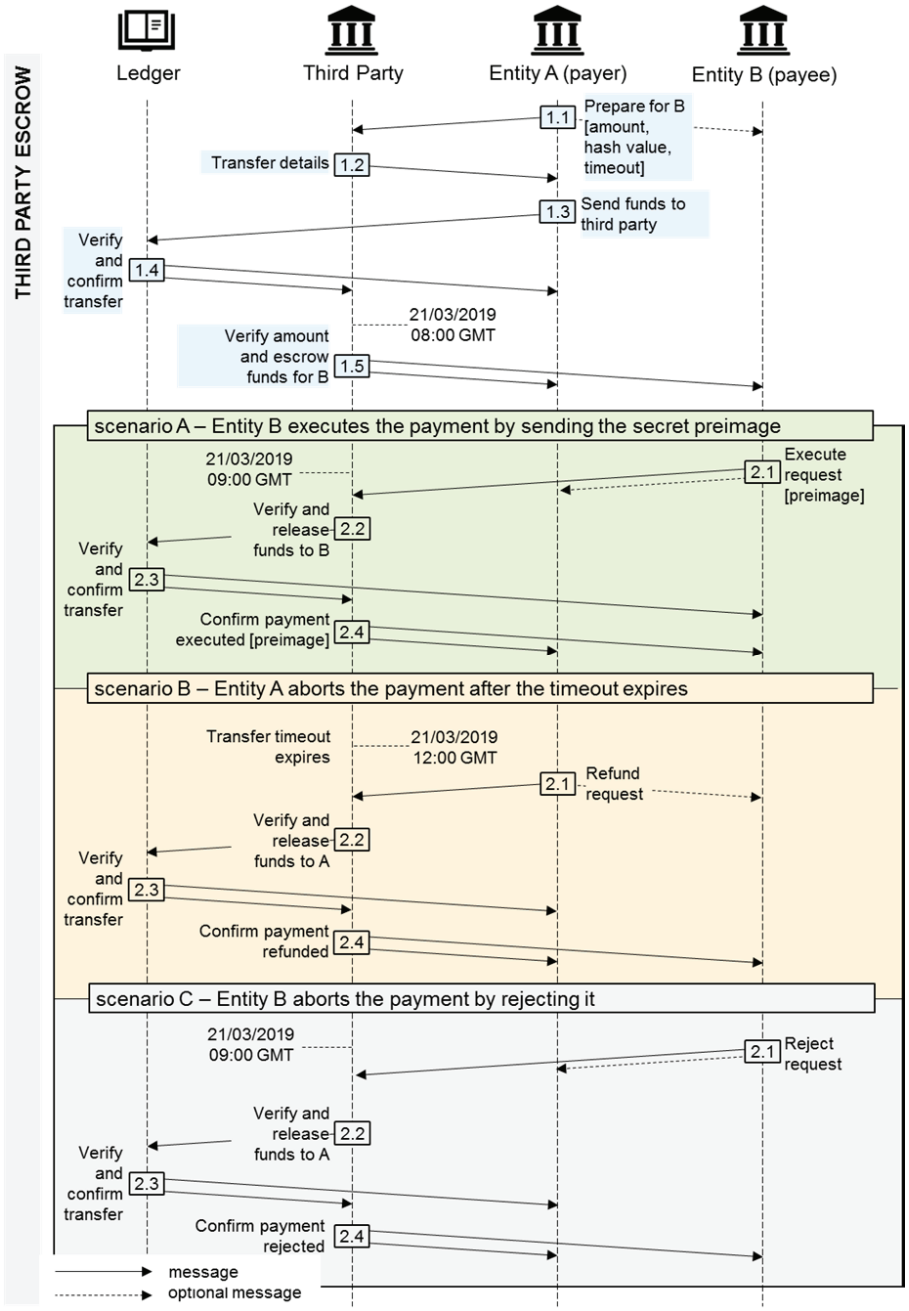
On-ledger escrow using HTLC process flow



Third party escrow

Third party escrow is conceptually similar to the on-ledger escrow using HTLC (see Figure C).

Figure C
Third party escrow process flow

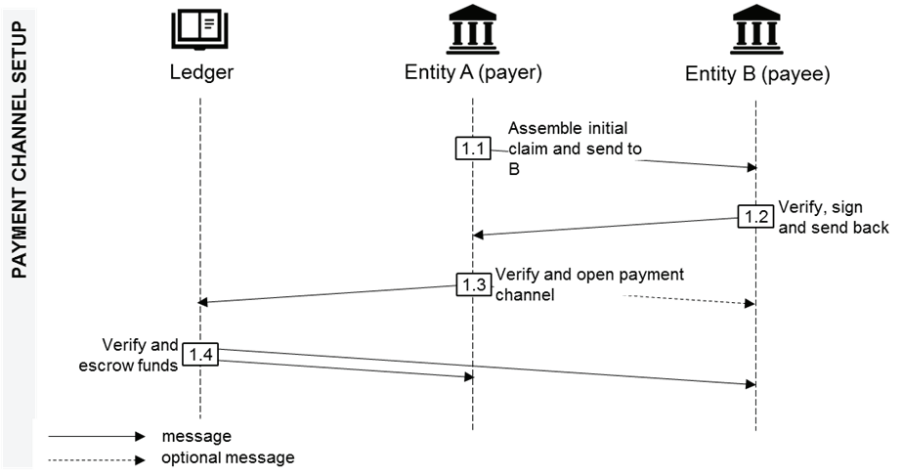


Funds are transferred to the third party during the preparation with a standard ledger transfer. The transfer details (e.g. the account identifier of the third party, the payment description, etc.) related to the payment are sent from the third party to the payer at step 1.2. Same considerations as on-ledger escrow using HTLC about the optional bidirectional communication between the entities apply for third party escrow.

Payment channels

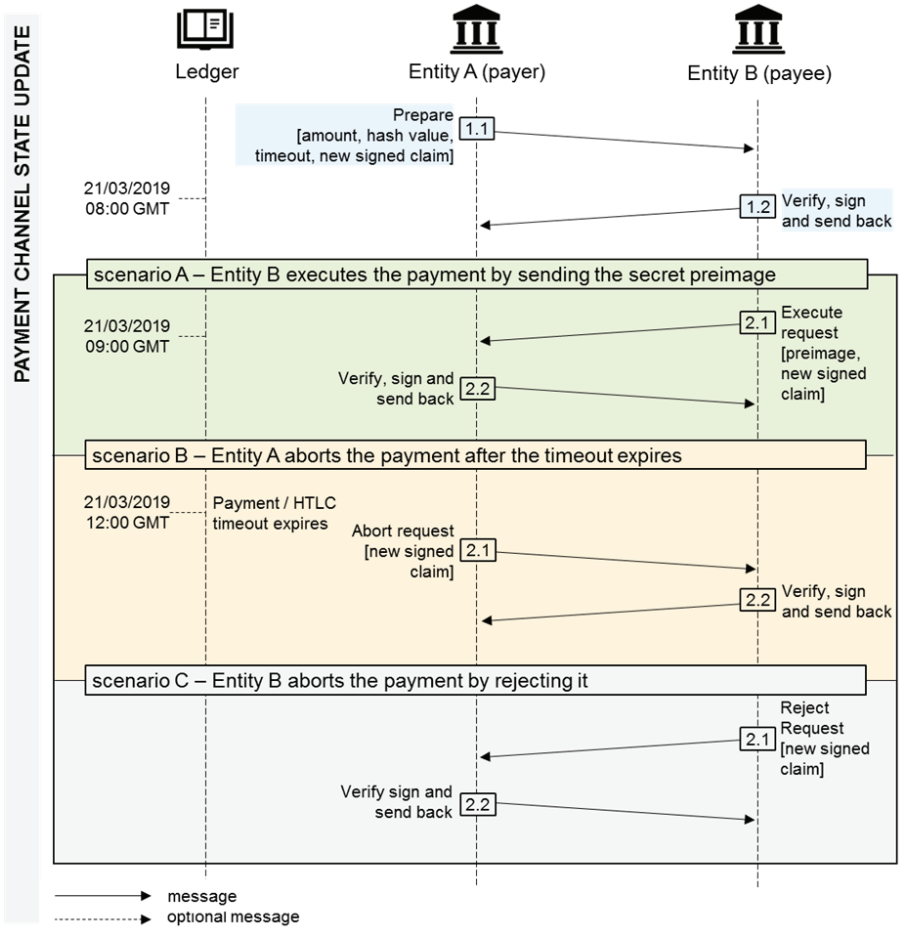
During the payment channel setup phase (see Figure D), the entity that funds the channel (in the figure Entity A) asks for a signed claim to the initial payment channel state from the counterparty (in the figure Entity B). In fact, once the payment channel is open and the funds are moved into the joint payment channel account, they are locked for an indefinite amount of time (payment channels considered in this report are not time-locked[62]) and both signatures by participants are required to release them. The initial claim exchange prevents a situation where the funding party (Entity A) escrows its funds in the payment channel and is then unable to release them because the counterparty (Entity B) does not cooperate. Once the signed claim has been sent to the funding party, the payment channel is opened with an on-ledger transaction.

Figure D
Payment channel setup phase



[62] Payment channels where the locked funds are released upon expiration of a timeout also exist, but are not considered in this report.

Figure E
Payment channel state update phase



In the payment channel state update phase the involved entities exchange signed claims to new payment channel states (see Figure E).

During the preparation phase (steps 1.1 and 1.2), funds are debited from the payer's balance in the payment channel state and credited to a prepared payment. In the case of simple payment channel, the prepared payment can be assigned to one of the counterparties (payer, payee, entity A regardless of the payer, entity B regardless of the payer) or a third party, depending on trust relationships between A and B. Within this report we choose a specific simple payment channel design where the payee trusts the payer and prepared funds are always assigned to the payer's balance in the claim, leading to unchanged balances during payment preparations.[63]

[63] In the specific design chosen in this report, there is no need to update the simple payment channel state during preparation, so a new claim is not exchanged in step 1.1.

In the case of conditional payment channel, a new HTLC for the prepared payment is added to the payment channel state.

In simple payment channel, the delivery of the promised claim in exchange for a hash preimage, the verification of the validity of that preimage and that of its timely submission is conducted off-ledger by the entity that handles the prepared payment. Within this report this entity is the payer.[64] In scenario A, the new claim debits the prepared amount from the payer (the entity which controls prepared payments) and credits the prepared amount to the payee.

In conditional payment channels, as long as the payment channel remains open, the verification of the fulfilment condition is made by the payer (scenario A) and the verification of the payment timeout is made by the payee (scenario B). In scenario A, the new claim credits the prepared amount to the payee in the payment channel state. In scenarios B and C, the new claim credits the prepared amount to the payer. In all of the above three scenarios, the HTLC added during the preparation phase is removed from the new payment channel state. It is also important to note that in the case of conditional payment channel, the enforcement of the fulfilment condition and payment timeout can be carried out by the ledger if the payment channel is closed, while in the case of simple payment channel this option is not present.

[64] In the specific design chosen in this report, there is no need to update the simple payment channel state during abort, so a new claim is not exchanged in scenarios B and C.

Figure F
Payment channel settlement phase

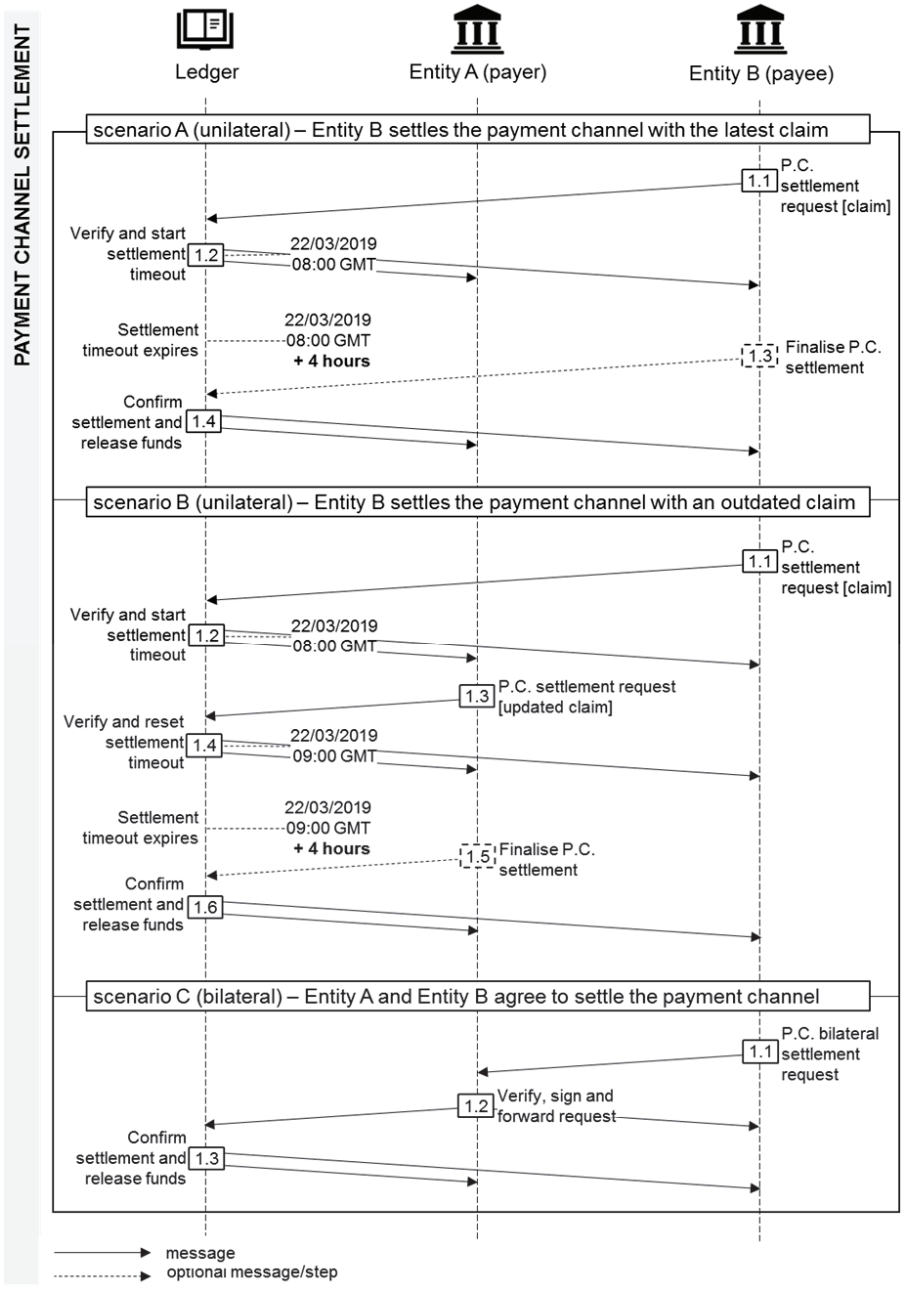


Figure F shows three possible scenarios for the payment channel settlement phase: the first two scenarios (A and B) are unilateral settlements of the payment channel, while the third one is a bilateral payment channel settlement. Unilateral payment channel settlement happens when one of the two participants unilaterally takes the initiative to start the settlement phase by sending a claim to the ledger. In this case,

closing the payment channel requires time, because the ledger operator must ensure that the received claim is actually a claim to the latest payment channel state that has been agreed between participants. Bilateral payment channel settlement (scenario C) happens when participants agree to start the settlement phase and co-sign a payment channel settlement transaction. In this case, the settlement phase only includes the settlement transaction to be confirmed on-ledger.

The sequence of payment channel states is known only to the participants and the ledger has no information about which claim is actually the one to the latest agreed payment channel state. It could happen that a participant willing to perform unilateral channel close sends to the ledger a claim to an outdated payment channel state (Entity B in scenario B, step 1.1). Before finalising the unilateral closing of the payment channel, the ledger has to wait for a settlement timeout in order to enforce the latest agreed payment channel state. Before the expiration of this timeout, the other participant has the option to react to the unilateral close by sending to the ledger a claim to an updated payment channel state (Entity A in scenario B, step 1.5). The ledger must have a mechanism, called *replacement mechanism*, to distinguish between an outdated claim and an updated one, and to enforce the settlement of the most updated claim if this is received before the expiration of the settlement timeout. Every time a new payment channel claim is sent to the ledger, the settlement timeout is reset (restarted). Only when the settlement timeout expires can participants finalise the payment channel settlement and the actual transfer of funds happens.

Ledger requirements for payment methods

In this section, some details are presented about the two functionalities for ledgers that were identified in Section 4.5, namely those for HTLC and payment channels. HTLC requirements are derived from procedures for lightning network HTLC[65], while payment channel requirements are derived from procedures for a payment channel proposal called eltoo[66].[67]

Since both HTLC and payment channels rely on timely actions by the participants, an important requirement for both functionalities is about time: the flow of time in the ledgers must be predictable by the participants in order for them to make assessment and decide for safe timeouts. These timeouts can be either absolute (like the ones in HTLC) or relative (like the ones in payment channels).

Another common requirement is about verifying user credentials. In advanced ledgers this is achieved using digital signatures.

Hashed Timelock Contracts requirements

1. During the preparation, on receipt of an HTLC prepare request (containing the amount, hash value, and timeout), the ledger must verify that the request is valid and comes from the payer, and the payer's account contains enough funds. Then the ledger must debit the payer's account and put funds in escrow or "hold state" until the expiration of the HTLC timeout (timelock). When funds are in the hold state, participants cannot move them freely: the payer cannot retrieve the funds before the defined timeout and the payee cannot receive them until it provides the valid hash preimage to the defined HTLC hash value. Also, the ledger must notify involved participants when the contract has been created.

2. After the preparation, on receipt of an HTLC execute request, the ledger must verify that the request is valid and comes from the payee, and the hash of the provided preimage matches with the hash value, and the timeout is not expired. Then the ledger has to credit the escrow funds to the payee's account. Finally, the ledger must notify participants that the condition has been fulfilled and communicate the preimage.

3. After the preparation, on receipt of an HTLC refund request, the ledger must verify that the request is valid and comes from the payer, and the timeout is expired. Then the ledger has to credit the escrow funds to the payer's account. Finally the ledger must notify participants that the timeout has expired and the HTLC has been refunded.

[65] https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md

[66] https://blockstream.com/eltoo.pdf

[67] Lightning network and eltoo are implementations based on Bitcoin. Therefore, the requirements raised in this section may not apply, or additional requirements may be necessary for implementations based on other platforms that are significantly different from Bitcoin.

4. (A requirement in the case where the ledger wants to provide the payee with the possibility to reject the payment[68]). After the preparation, on receipt of an HTLC reject request, the ledger must verify that the request is valid and comes from the payee. Then the ledger has to credit the escrow funds to the payer's account. Finally the ledger must notify participants that the HTLC has been rejected.

As a summary of the above requirements:

- The ledger must be able to put funds in escrow until predefined timeout.
- The ledger must be able to verify the hash value of the provided preimage matches with the predefined hash value in the contract and the expiration of predefined timeout.
- The ledger must credit the escrow funds to the payee's account if the preimage is verified successfully and timeout is not expired.
- The ledger must credit the escrow funds to the payer's account when the timeout is expired.
- The ledger must be able to notify involved participants if these events happen.
    - The contract has been created.
    - The condition has been fulfilled. In addition, the ledger must communicate the preimage.
    - The timeout has expired and the HTLC has been refunded.

Payment Channel requirements

1. In the setup phase, on receipt of a payment channel open request, the ledger must verify that the request is valid and comes from the entity funding the channel, and the funder's account contains enough money. Then the ledger must debit the funder's account and put funds into a shared escrow account until the receipt of a claim that is signed by both participants and requests the payment channel settlement. Finally, the ledger must notify involved participants that the payment channel has been created and the terms of it.

2. In the state update phase, on receipt of a payment channel claim, the ledger must verify that the claim is valid and has been signed by both participants. The payment channel claim includes a transaction for unilateral settlement. Then the ledger must start the unilateral settlement phase. Since the channel is going to be unilaterally closed, the settlement timeout must be started. Finally, the ledger must notify channel participants that the settlement phase has started and must communicate them the candidate final payment channel state.

[68] Rejection was not part of the initial proposal introduced in the whitepaper, where the payment chain can either be executed or aborted. While it may be possible for the payee to reject a transfer rather than to wait for the timeout, there is no economic incentive for the payee to do so.

3. In the unilateral settlement phase, the ledger must allow participants to commit the most recently updated payment channel state and solve the disputes about claims between the participants of the payment channel. On receipt of a valid and co-signed claim to an updated payment channel state the ledger must replace the candidate final payment channel state with the updated one. Then the ledger must reset (restart) the settlement timeout. Finally the ledger must notify channel participants that the candidate final payment channel state has been replaced and must communicate the new candidate final payment channel state.

4. In the unilateral settlement phase, on expiration of the settlement timeout, the ledger must allow participants to finalise settlement and redistribute the funds in the shared account according to the positions contained in the final payment channel state that was sent within the timeout. Finally the ledger must notify participants that the settlement phase has ended and the payment channel has been closed.

5. In the state update phase, on receipt of a bilateral channel close request, the ledger must check that the request has been co-signed by both participants and must redistribute funds in the shared account according to the positions contained in the bilateral channel close request. Finally the ledger must notify the participants that the payment channel has been closed.

As a summary of the above requirements:

- The ledger must be able to put funds into a shared escrow account until the receipt of a claim that is signed by both participants.
- The ledger must be able to replace the candidate final payment channel state with the updated one.
- The ledger must be able to allow participants to finalise settlement and redistribute the funds in the shared account according to the positions contained in the final payment channel state that was sent within the timeout.
- The ledger must be able to redistribute funds in the shared account according to the positions contained in the bilateral channel close request.
- The ledger must be able to notify involved participants about the term of the events described below when any one of them happens.
    - The payment channel has been created or closed.
    - The settlement phase has started or finished.

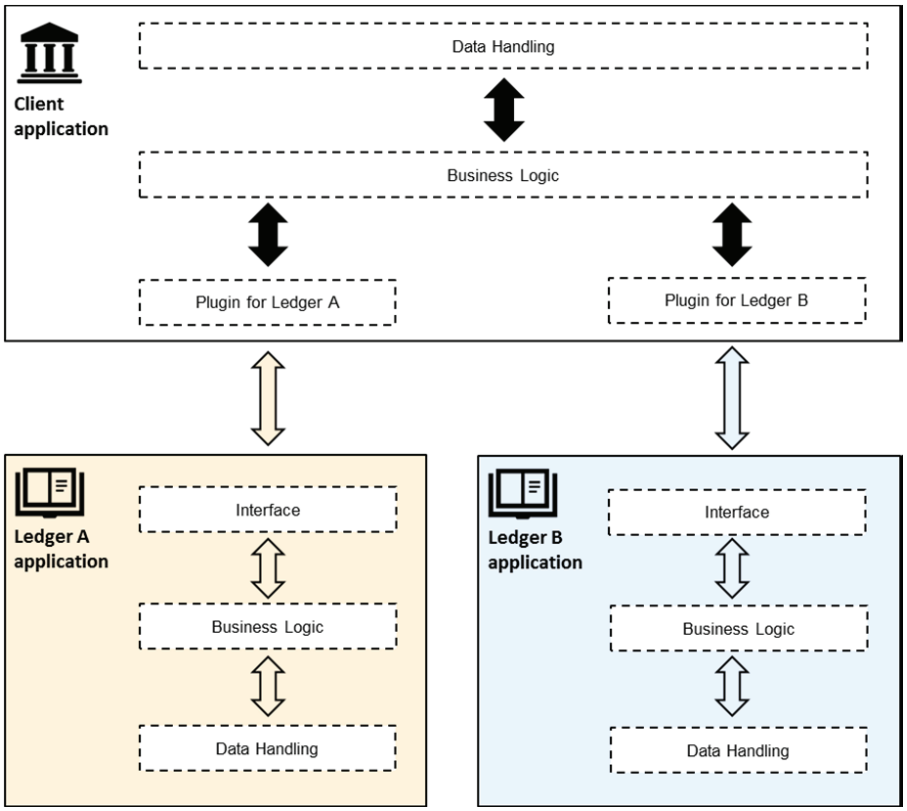
Annex 2: Details of experiments

To carry out experiments of synchronised payments, ledgers and client applications for participants are required. This chapter shows the details of the experiment setup.

Plugin architecture

Plugin architecture is said to be widely adopted in funds transfers using ILP to abstract the differences between ledgers and client applications. This experiment also adopted plugin architecture for transfers when using ILP, as it dealt with funds transfers between ledgers on different interfaces. The next figure shows the design of a client application used in the experiment. Arrows indicate the information exchange between modules. Arrows of the same colour indicate message exchanges according to the same platform-specific format.

Figure G
Plugin architecture



Validation of hash value

The hash values in the experiments with ILP were calculated according to the Pre-Shared Key Transport Protocol (PSK).[69]

In PSK, the Receiver shares a secret (called shared secret) with the Sender before initiating the funds transfer. The Sender calculates the hash preimage from the terms of the transfer (e.g. destination amount and account) and the shared secret by using HMAC. Finally the Sender calculates the payment condition (also called hash values in Chapter 3) from the hash preimage by using SHA256.[70] To enable serialisation, the hash value and the payment condition are encoded/decoded to Base64 format during the calculation.

In these experiments, the possession of the hash preimage to the payment condition by the Sender cannot be considered as a non-repudiable proof of payment because it is already known to the Sender when the transfer is initiated.

Experiment on centralised ledger

For the ledger, Five Bells Ledger was used.[71] For the application of the participants (the Sender, Connector, Receiver), ilp-plugin-bells[72] was used. In addition, for the application of the Connector, ilp-connector[73] was used, and for the application of the Sender and Receiver, ilp[74] was used to generate, to verify and to manage a hash value and a preimage for the transfer.[75] The environment used in the experiment on centralised ledgers is as below.

[69] https://interledger.org/rfcs/0016-pre-shared-key/

[70] The preimage is the output of HMAC (HMAC (secret, “ilp_psk_condition”), base64decode(packet)). HMAC is Hash-based Message Authentication Code.

[71] Version 21.2.5 was used.

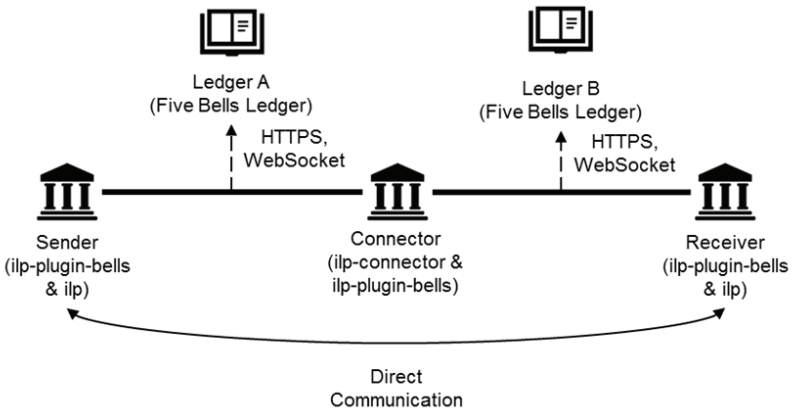
[72] Version 15.1.2 was used.

[73] Version 21.1.10 was used.

[74] Different from "ILP", "ilp" in lower case is an application in Interledger.js, including some protocols of Interledger protocol stack. ilp version 11.4.0 was used.

[75] The deprecated Interledger protocol suite used in the experiment comprises four layers: the Application, Transport, Interledger, and Ledger protocols. Of these, the Application layer and the Ledger layer were developed and implemented in the experiments to understand the process these layers are responsible for.

Figure H
Centralised ledgers experiment

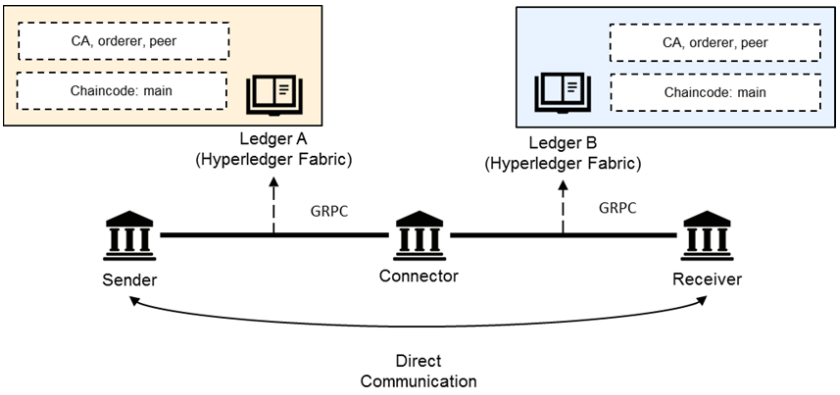


Experiments on DLT ledgers

Experiment without ILP

We developed several chaincodes[76] – including those to open an account, transfer funds in the same ledger, escrow, release, and expire – and we installed them on both Ledgers. The experiments were carried out by executing chaincodes (named "main"), in the order shown in Figure I.

Figure I
DLT ledgers experiment without ILP



[76] A "chaincode" in Hyperledger Fabric refers to a smart contract. These functionalities were similar to the ones implemented in Five Bells Ledger.

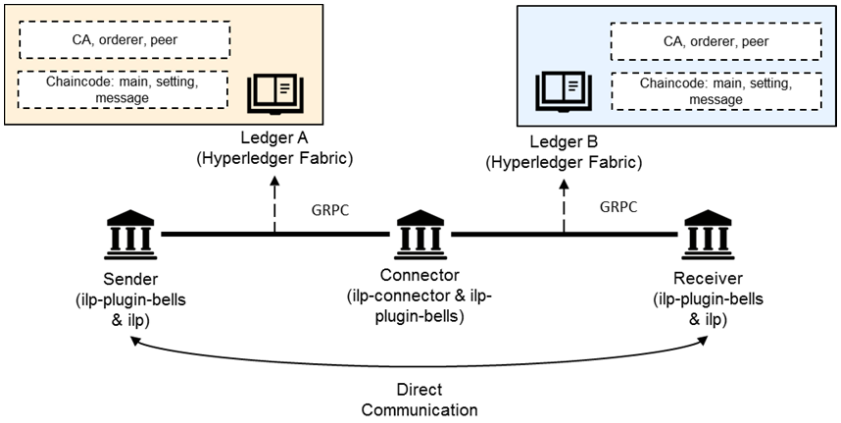
Experiment with ILP

We carried out the following tasks:

1. Implement a chaincode which helps message passing between participants, named “message”
    - Notify the terms of transfer to the payee when an on-ledger escrow is created.
    - Notify the preimage to the payer when escrowed funds are transferred after the preimage is presented by the payee.
    - Notify the quoting request from the Sender to the Connector.
    - Notify the response of the quoting request from the Connector to the Sender.[77]
2. Added a chaincode which references the information on the ledger, named “setting”
    - Return the information on the ledger (e.g. currency code and currency scale of the client account).
3. Extension of the main chaincode
    - Record all relevant terms of the transfer (e.g. destination, amount and address). In the original implementation, the ledger recorded information on the hash value, payment amount, payment destination, and payment timeout.
    - Change the specification method of timeout from relative time to absolute time.
    - Adoption of the same hash function as ilp.

[77] Message passing functionalities are specific to ILPv3.

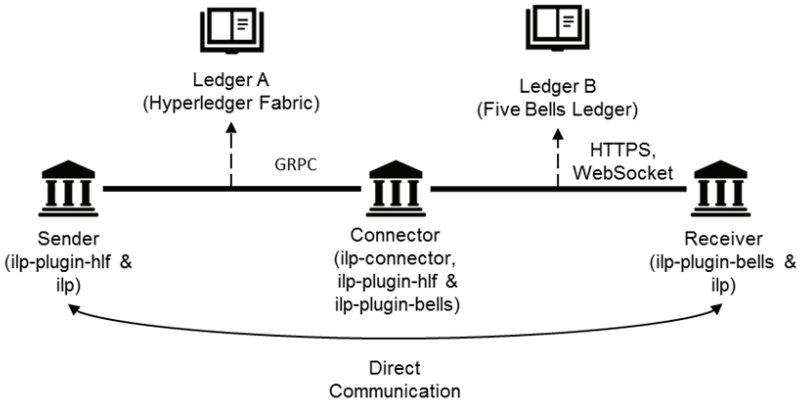
Figure J
DLT ledgers experiment with ILP



Cross-platform experiment

The client application of the Sender and the Receiver was not changed, since both client applications are ILPv3 compliant. After the addition of the plugin to the connector client application, the procedure of the transfer is not changed.

Figure K
Cross-platform experiment with ILP



Postal address 60640 Frankfurt am Main, Germany
Telephone +49 69 1344 0
Website www.ecb.europa.eu

Postal address 2-1-1 Nihonbashi-Hongokucho, Chuo-ku, Tokyo 103-0021, Japan
Telephone +81 3 3279-1111
Website www.boj.or.jp